

ALGORITMA STRUKTUR DATA
Praktikum – Double Linked List
Lavina 2341760062

Praktikum 1: Pembuatan Linked List

Node.java

```
1  package doublelinkedlists;
2
3  /**
4   * Node
5   */
6  public class Node {
7
8      int data;
9      Node prev, next;
10
11     Node(Node prev, int data, Node next) {
12         this.prev = prev;
13         this.data = data;
14         this.next = next;
15     }
16 }
```

DoubleLinkedLists.java

```
1  package doublelinkedlists;
2
3  public class DoubleLinkedLists {
4      Node head;
5      int size;
6
7      public DoubleLinkedLists() {
8          head = null;
9          size = 0;
10     }
11 }
```

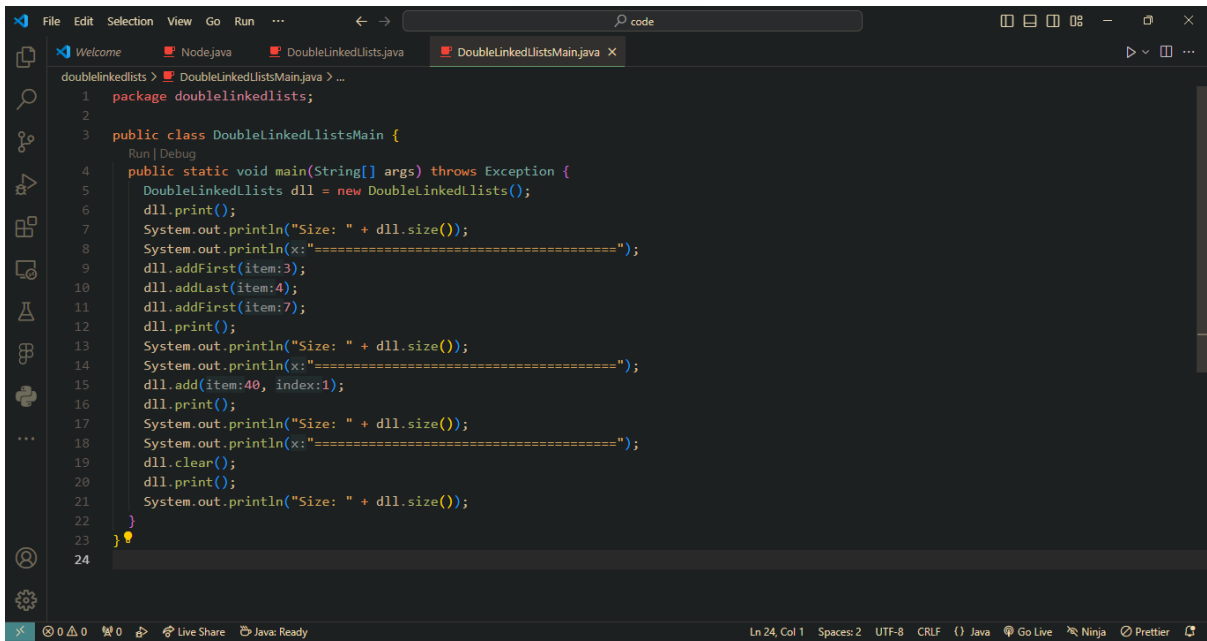
```
12     public int size() {
13         return size;
14     }
15
16     public boolean isEmpty() {
17         return head == null;
18     }
19
20     public void clear() {
21         head = null;
22         size = 0;
23     }
24
25     public void addFirst(int item) {
26         if (isEmpty()) {
27             head = new Node(null, item, null);
28         } else {
29             Node newNode = new Node(null, item, head);
30             head.prev = newNode;
31             head = newNode;
32         }
33         size++;
34     }
35
36     public void addLast(int item) {
37         if (isEmpty()) {
38             addFirst(item);
39         } else {
40             Node current = head;
41             while (current.next != null) {
42                 current = current.next;
43             }
44             Node newNode = new Node(current, item, null);
45             current.next = newNode;
46             size++;
47         }
48     }
49
50     public void add(int item, int index) throws Exception {
51         if (isEmpty()) {
52             addFirst(item);
```

```

53     } else if (index < 0 || index > size) {
54         throw new Exception("Nilai index diluar batas");
55     } else {
56         Node current = head;
57         int i = 0;
58         while (i < index) {
59             current = current.next;
60             i++;
61         }
62         if (current.prev == null) {
63             Node newNode = new Node(null, item, current);
64             current.prev = newNode;
65             head = newNode;
66         } else {
67             Node newnNode = new Node(current.prev, item, current);
68             newnNode.prev = current.prev;
69             newnNode.next = current;
70             current.prev.next = newnNode;
71             current.prev = newnNode;
72         }
73     }
74     size++;
75 }
76
77 public void print() {
78     if (!isEmpty()) {
79         Node tmp = head;
80         while (tmp != null) {
81             System.out.print(tmp.data + "\t");
82             tmp = tmp.next;
83         }
84         System.out.println("\nberhasil diisi");
85     } else {
86         System.out.println("Linked list kosong");
87     }
88 }
89 }

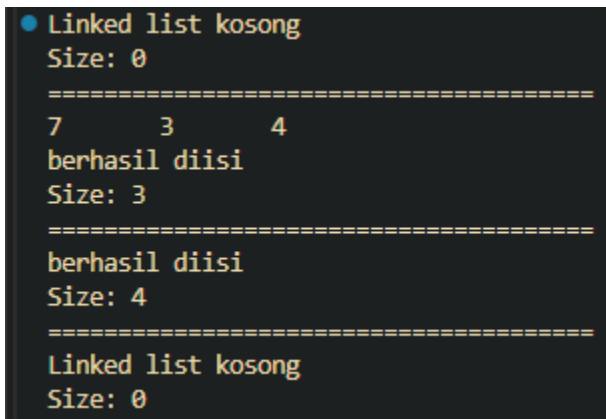
```

DoubleLinkedListsMain.java



```
1 package doublelinkedlists;
2
3 public class DoubleLinkedListsMain {
4     public static void main(String[] args) throws Exception {
5         DoubleLinkedLists dll = new DoubleLinkedLists();
6         dll.print();
7         System.out.println("Size: " + dll.size());
8         System.out.println(x:"=====");
9         dll.addFirst(item:3);
10        dll.addLast(item:4);
11        dll.addFirst(item:7);
12        dll.print();
13        System.out.println("Size: " + dll.size());
14        System.out.println(x:"=====");
15        dll.add(item:40, index:1);
16        dll.print();
17        System.out.println("Size: " + dll.size());
18        System.out.println(x:"=====");
19        dll.clear();
20        dll.print();
21        System.out.println("Size: " + dll.size());
22    }
23 }
24
```

Output :



```
• Linked list kosong
Size: 0
=====
7      3      4
berhasil diisi
Size: 3
=====
berhasil diisi
Size: 4
=====
Linked list kosong
Size: 0
```

Pertanyaan

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

Jawab : Double linked list memiliki 2 pointer yaitu prev dan next, sedangkan single linked list hanya punya 1 pointer next.

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

Jawab : Atribut next untuk pointer yang akan menunjuk ke node berikutnya dari node saat ini, sedangkan atribut prev untuk pointer ke node sebelumnya dari node saat ini.

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

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

Jawab : Karena dengan konstruktor yang merupakan method yang akan dieksekusi pertama sehingga bisa menginisialisasi kondisi awal dari linked list menjadi kosong dan node pertamanya null.

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

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

Jawab : Karena method ini bertujuan untuk menambahkan data baru diposisi head, dan posisi head pointer prev nya menunjuk ke null.

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

Jawab : Artinya data yang baru dimasukkan ke posisi sebelum head.

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);
```

Jawab : Dibuat demikian karena method tersebut berfungsi untuk memasukkan data diakhir linked list, diakhir link list pointer prev harus diarahkan ke current node dan pointer next nya harus menunjuk ke null.

7. Pada method add(), terdapat potongan kode program sebagai berikut:

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

jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

Jawab : Maksudnya adalah jika node sebelum dari node saat ini (current) kosong, yang berarti merupakan head dari linked list maka node yang baru akan diletakkan diposisi head tersebut.

Praktikum 2

DoubleLinkedLists.java

```
1  public void removeFirst() throws Exception {
2      if (isEmpty()) {
3          throw new Exception("Linked list masih kosong, tidak dapat dihapus");
4      } else if (size == 1) {
5          removeLast();
6      } else {
7          head = head.next;
8          head.prev = null;
9          size--;
10     }
11 }
12
13 public void removeLast() throws Exception {
14     if (isEmpty()) {
15         throw new Exception("Linked list masih kosong, tidak dapat dihapus");
16     } else if (head.next == null) {
17         head = null;
18         size--;
19         return;
20     }
21
22     Node current = head;
23     while (current.next.next != null) {
24         current = current.next;
25     }
26     current.next = null;
27     size--;
28 }
29
30 public void remove(int index) throws Exception {
31     if (isEmpty() || index >= size) {
32         throw new Exception("Nilai index diluar batas");
33     } else if (index == 0) {
34         removeFirst();
35     } else {
36         Node current = head;
37         int i = 0;
38         while (i < index) {
```

```

39     current = current.next;
40     i++;
41 }
42 if (current.next == null) {
43     current.prev.next = null;
44 } else if (current.prev == null) {
45     current = current.next;
46     current.prev = null;
47     head = current;
48 } else {
49     current.prev.next = current.next;
50     current.next.prev = current.prev;
51 }
52 size--;
53 }
54 }

```

DoubleLinkedListsMain.java

```

3 public class DoubleLinkedListsMain {
4     public static void main(String[] args) throws Exception {
23         dll.addLast(item:50);
24         dll.addLast(item:40);
25         dll.addLast(item:10);
26         dll.addLast(item:20);
27         dll.print();
28         System.out.println("Size: " + dll.size());
29         System.out.println(x:"=====");
30         dll.removeFirst();
31         dll.print();
32         System.out.println("Size: " + dll.size());
33         System.out.println(x:"=====");
34         dll.removeLast();
35         dll.print();
36         System.out.println("Size: " + dll.size());
37         System.out.println(x:"=====");
38         dll.remove(index:1);
39         dll.print();
40         System.out.println("Size: " + dll.size());
41     }
42 }
43

```

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

```

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

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

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

Jawab : Data yang sebelumnya menjadi head diubah menjadi data yang ada disetelahnya, lalu data yang sebelumnya head diubah menjadi null sehingga data yang sebelumnya menjadi head akan terhapus.

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

Jawab : Caranya dengan melakukan loop untuk mengecek setiap data next dan nextnya lagi current null, seperti yang ditunjukkan pada kode berikut :

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

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

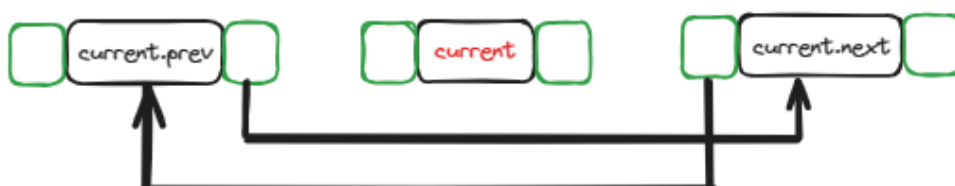
```
Node tmp = head.next;  
head.next=tmp.next;  
tmp.next.prev=head;
```

Jawab : Fungsi tersebut hanya mengubah susunan dari linked list dengan mengabaikan node kedua, tetapi node kedua masih ada di memori, data dari node tersebut masih ada hanya saja tidak dapat diakses.

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

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

Jawab : Fungsinya adalah untuk menghapus data current dengan cara yang aman, cara kerjanya yaitu dengan mengarahkan pointer next dari node sebelum node current ke node setelah current, lalu node setelah current pointer prev nya diarahkan ke node sebelum node saat ini. Berikut gambarannya :



Praktikum 3

DoubleLinkedLists.java

```
1    public int getFirst() throws Exception {
2        if (isEmpty()) {
3            throw new Exception("Linked list kosong");
4        }
5        return head.data;
6    }
7
8    public int getLast() throws Exception {
9        if (isEmpty()) {
10            throw new Exception("Linked list kosong");
11        }
12        Node tmp = head;
13        while (tmp.next != null) {
14            tmp = tmp.next;
15        }
16        return tmp.data;
17    }
18
19    public int get(int index) throws Exception {
20        if (isEmpty() || index >= size) {
21            throw new Exception("Nilai index diluar batas.");
22        }
23        Node tmp = head;
24        for (int i = 0; i < index; i++) {
25            tmp = tmp.next;
26        }
27        return tmp.data;
28    }
29
```

DoubleLinkedListsMain.java

```
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"=====");
System.out.println("Data awal pada linked list adalah: " + dll.getFirst());
System.out.println("Data akhir pada linked list adalah: " + dll.getLast());
System.out.println("Data index ke-1 pada linked list adalah: " + dll.get(index:1));
```

Output :

```
Linked list kosong
Size: 0
=====
7      3      4
berhasil diisi
Size: 3
=====
7      40     3      4
berhasil diisi
Size: 4
=====
7      40     3      4
berhasil diisi
Size: 4
=====
Data awal pada linked list adalah: 7
Data akhir pada linked list adalah: 4
Data index ke-1 pada linked list adalah: 40
```

Pertanyaan

1. Jelaskan method size() pada class DoubleLinkedLists!

Jawab : Method ini berfungsi untuk mendapatkan nilai dari jumlah data yang ada di linked list dan memperbarui jumlahnya secara otomatis;

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!

Jawab : Bisa dengan menambahkan node diawal dan diakhir sebagai pembatas, tidak perlu diisi datanya, sehingga undexnya bisa dimulai dari 1.

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

Jawab : Pada DLL saat melakukan add pointer yang harus diubah posisinya ada 2 yaitu pointer prev dan next, sedangkan SLL hanya ada 1 yaitu 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)

Jawab : Pada method A mengecek size dari linked list apakah kosong atau tidak, sedangkan pada method yang B yang dicek adalah headnya, jika null maka linked listnya kosong.

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)

Ilustrasi Program

Menu Awal dan Penambahan Data

```
+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
-++++
```

```
+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
1
-----
Masukkan Data Penerima Vaksin
-----

Nomor Antrian:
123
-Nama Penerima:
Joko
```

Cetak Data (Komponen di area merah harus ada)

```
+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
3
+++++
Daftar Pengantri Vaksin
+++++
|No.   |Nama  |
|123   |Joko  |
|124   |Mely  |
|135   |Johan |
|146   |Rosi  |
|sisa Antrian: 4
```

Hapus Data (Komponen di area merah harus ada)

```
*****
PENGANTRI VAKSIN EXTRAVAGANZA
*****

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

2
Joko telah selesai divaksinasi.
*****
Daftar Pengantri Vaksin
*****
|No.   |Nama |
|124   |Mely |
|135   |Johan|
|146   |Rosi |
Sisa Antrian: 3
```

Node.java

```
1  package vaksin;
2
3  public class Node {
4
5      String id, nama;
6      Node prev, next;
7
8      Node(Node prev, String id, String nama, Node next) {
9          this.prev = prev;
10         this.id = id;
11         this.nama = nama;
12         this.next = next;
13     }
14
15     Node(String id, String nama) {
16         this.id = id;
17         this.nama = nama;
18     }
19 }
```

DoubleLinkedLists.java

```
1  package vaksin;
2
3  public class DoubleLinkedLists {
4      Node head;
5      int size;
6
7      public DoubleLinkedLists() {
8          head = null;
9          size = 0;
10     }
11
12     public int size() {
13         return size;
14     }
15
16     public boolean isEmpty() {
17         return head == null;
18     }
19
20     public void addFirst(String id, String nama) {
21         if (isEmpty()) {
22             head = new Node(null, id, nama, null);
23         } else {
24             Node newNode = new Node(null, id, nama, head);
25             head.prev = newNode;
26             head = newNode;
27         }
28         size++;
29     }
30
31     public void addLast(String id, String nama) {
32         if (isEmpty()) {
33             addFirst(id, nama);
34         } else {
35             Node current = head;
36             while (current.next != null) {
37                 current = current.next;
38             }
39             Node newNode = new Node(current, id, nama, null);
40             current.next = newNode;
41             size++;
42         }
43     }
44
45     public void removeFirst() throws Exception {
46         if (isEmpty()) {
47             throw new Exception("Linked list masih kosong, tidak dapat dihapus");
48         } else if (size == 1) {
49             removeLast();
50         } else {
51             System.out.println(head.nama + " telah selesai divaksinasi.");
52             head = head.next;
53             head.prev = null;
54             size--;
```

```

55     }
56 }
57
58 public void removeLast() throws Exception {
59     if (isEmpty()) {
60         throw new Exception("Linked list masih kosong, tidak dapat dihapus");
61     } else if (head.next == null) {
62         System.out.println(head.nama + " telah selesai divaksinasi.");
63         head = null;
64         size--;
65         return;
66     }
67
68     Node current = head;
69     while (current.next.next != null) {
70         current = current.next;
71     }
72     current.next = null;
73     size--;
74 }
75
76 public void print() {
77     System.out.println("+++++++");
78     System.out.println("Daftar Pengantri Vaksin");
79     System.out.println("+++++++");
80     System.out.println("| No.\t| Nama\t|");
81     if (!isEmpty()) {
82         Node tmp = head;
83         while (tmp != null) {
84             String id = tmp.id;
85             String nama = tmp.nama;
86             String format = "| %-5s | %-5s |";
87             System.out.println(String.format(format, id, nama));
88             tmp = tmp.next;
89         }
90         System.out.println("\nSisa Antrian: " + size);
91     } else {
92         System.out.println("Linked list kosong");
93     }
94 }
95 }

```

DoubleLinkedListsMain.java

```

1  package vaksin;
2
3  import java.util.Scanner;
4
5  public class DoubleLinkedListsMain {
6      public static void main(String[] args) throws Exception {
7          Scanner sc = new Scanner(System.in);
8          DoubleLinkedLists dll = new DoubleLinkedLists();
9
10         int pilih;
11         do {
12             System.out.println("+++++++");
13             System.out.println("PENGANTRI VAKSIN EXTRAVAGANZA");
14             System.out.println("+++++++");
15             System.out.println(
16                 "\n1. Tambah data penerima vaksin\n2. Hapus data pengantri vaksin\n3. Daftar penerima vaksin\n4. Keluar");
17             System.out.println("+++++++");
18             pilih = sc.nextInt();

```

```

19     sc.nextLine();
20
21     switch (pilih) {
22         case 1:
23             System.out.println("=====");
24             System.out.println("Masukkan Data Penerima Vaksin");
25             System.out.println("=====");
26             System.out.println("Nomor Antrian: ");
27             String id = sc.nextLine();
28             System.out.println("Nama Penerima: ");
29             String nama = sc.nextLine();
30             Node nd = new Node(id, nama);
31             dll.addLast(id, nama);
32             break;
33         case 2:
34             dll.removeFirst();
35             break;
36         case 3:
37             dll.print();
38             break;
39         case 4:
40             break;
41     }
42     } while (pilih < 4);
43 }
44 }

```

Output :

```

+++++
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

```

```

+++++
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

```

```

+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah data penerima vaksin
2. Hapus data pengantri vaksin
3. Daftar penerima vaksin
4. Keluar
+++++
2
Joko telah selesai divaksinasi.

```

```

+++++
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  |
| 124  | Mely  |
| 135  | Johan |
| 146  | Rosi  |

Sisa Antrian: 3

```

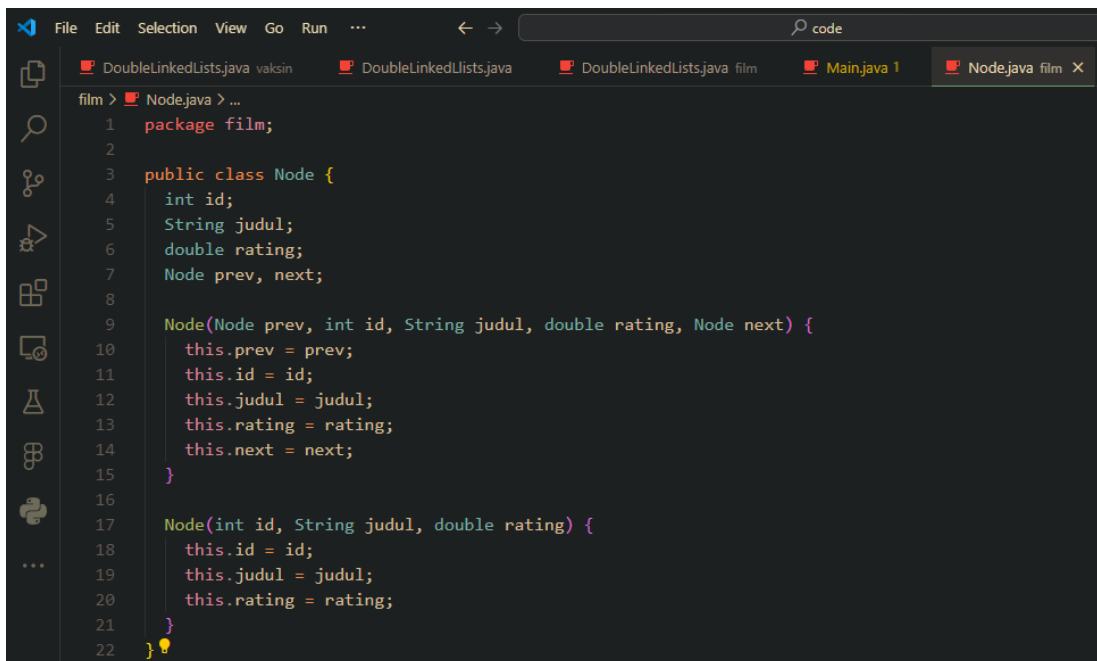
- ## Menu Awal dan Penambahan Data

```

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
=====
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

```


Node.java



```
1 package film;
2
3 public class Node {
4     int id;
5     String judul;
6     double rating;
7     Node prev, next;
8
9     Node(Node prev, int id, String judul, double rating, Node next) {
10         this.prev = prev;
11         this.id = id;
12         this.judul = judul;
13         this.rating = rating;
14         this.next = next;
15     }
16
17     Node(int id, String judul, double rating) {
18         this.id = id;
19         this.judul = judul;
20         this.rating = rating;
21     }
22 }
```

DoubleLinkedLists.java



```
1 package film;
2
3 public class DoubleLinkedLists {
4     Node head;
5     int size;
6
7     public DoubleLinkedLists() {
8         head = null;
9     }
10
11     public boolean isEmpty() {
12         return head == null;
13     }
14
15     public int size() {
16         return size;
17     }
18
19     public void addFirst(int id, String judul, double rating) {
20         if (isEmpty()) {
21             head = new Node(null, id, judul, rating, null);
22         } else {
23             Node newNode = new Node(null, id, judul, rating, head);
24             head.prev = newNode;
25             head = newNode;
26         }
27         size++;
28     }
29
30     public void addLast(int id, String judul, double rating) {
31         if (isEmpty()) {
32             addFirst(id, judul, rating);
33         } else {
```

```

34     Node current = head;
35     while (current.next != null) {
36         current = current.next;
37     }
38     Node newNode = new Node(current, id, judul, rating, null);
39     current.next = newNode;
40 }
41 size++;
42 }
43
44 public void add(int index, int id, String judul, double rating) {
45     if (isEmpty()) {
46         addFirst(id, judul, rating);
47     } else if (index < 0 || index > size) {
48         System.out.println("Index tidak valid !");
49     } else {
50         Node current = head;
51         int i = 0;
52         while (i < index) {
53             current = current.next;
54             i++;
55         }
56         if (current.prev == null) {
57             Node newNode = new Node(null, id, judul, rating, current);
58             current.prev = newNode;
59             head = newNode;
60         } else {
61             Node newNode = new Node(current.prev, id, judul, rating, current);
62             if (current.prev != null) {
63                 current.prev.next = newNode;
64             }
65             newNode.prev = current.prev;
66             newNode.next = current;
67             current.prev = newNode;
68         }
69     }
70 }
71 size++;
72 }
73
74 public void removeFirst() throws Exception {
75     if (isEmpty()) {
76         throw new Exception("Linked list masih kosong, tidak dapat dihapus");
77     } else if (size == 1) {
78         removeLast();
79     } else {
80         head = head.next;
81         head.prev = null;
82         size--;
83     }
84 }
85
86 public void removeLast() throws Exception {
87     if (isEmpty()) {
88         throw new Exception("Linked list masih kosong, tidak dapat dihapus");
89     } else if (head.next == null) {
90         head = null;
91         size--;
92         return;
93     }
94
95     Node current = head;
96     while (current.next.next != null) {
97         current = current.next;

```

```

98     }
99     current.next = null;
100    size--;
101 }
102
103 public void remove(int index) throws Exception {
104     if (isEmpty() || index >= size) {
105         throw new Exception("Nilai index diluar batas");
106     } else if (index == 0) {
107         removeFirst();
108     } else {
109         Node current = head;
110         int i = 0;
111         while (i < index) {
112             current = current.next;
113             i++;
114         }
115         if (current.next == null) {
116             current.prev.next = null;
117         } else if (current.prev == null) {
118             current = current.next;
119             current.prev = null;
120             head = current;
121         } else {
122             current.prev.next = current.next;
123             current.next.prev = current.prev;
124         }
125         size--;
126     }
127 }
128
129 public void print() {
130     System.out.println("Cetak Data");
131     if (!isEmpty()) {
132         Node data = head;
133         while (data != null) {
134             System.out.println("ID: " + data.id);
135             System.out.println(" Judul Film: " + data.judul);
136             System.out.println(" Rating: " + data.rating);
137             data = data.next;
138         }
139     } else {
140         System.out.println("Linked list kosong");
141     }
142 }
143
144 public void getId(int id) {
145     if (isEmpty()) {
146         System.out.println("Linked list kosong");
147     }
148
149     Node tmp = head;
150     int index = 0;
151     while (tmp != null && tmp.id != id) {
152         tmp = tmp.next;
153         index++;
154     }
155
156     if (tmp != null) {
157         System.out.println("Data ID Film: " + id + " berada di node ke-" + (index + 1));
158         System.out.println("IDENTITAS:");
159         System.out.println(" ID: " + tmp.id);
160         System.out.println(" Judul Film: " + tmp.judul);
161         System.out.println(" Rating: " + tmp.rating);

```

```

162     } else {
163         System.out.println("Film dengan ID: " + id + " tidak tersedia");
164     }
165 }
166
167 public void sortRating() {
168     if (head == null || head.next == null) {
169         return;
170     }
171
172     boolean swapped;
173     do {
174         swapped = false;
175         Node prev = null;
176         Node current = head;
177         Node nextNode = head.next;
178
179         while (nextNode != null) {
180             if (current.rating < nextNode.rating) {
181                 if (prev != null) {
182                     prev.next = nextNode;
183                 } else {
184                     head = nextNode;
185                 }
186
187                 current.next = nextNode.next;
188                 nextNode.next = current;
189                 prev = nextNode;
190                 nextNode = current.next;
191                 swapped = true;
192             } else {
193                 prev = current;
194                 current = nextNode;
195                 nextNode = nextNode.next;
196             }
197         }
198     } while (swapped);
199 }
200
201 }

```

Main.java

```

1  package film;
2
3  import java.util.Scanner;
4
5  public class Main {
6      public static void main(String[] args) throws Exception {
7          Scanner sc = new Scanner(System.in);
8          DoubleLinkedLists dll = new DoubleLinkedLists();
9
10         int pilih;
11         do {
12             System.out.println("=====");
13             System.out.println("DATA FILM LAYAR LEBAR");
14             System.out.println("=====");
15             System.out.println("1. Tambah Data Awal\n2. Tambah Data Akhir\n3. Tambah Data Index Tertentu");
16             System.out.println("4. Hapus Data Pertama\n5. Hapus Data Terakhir\n6. Hapus Data Tertentu");
17             System.out.println("7. Cetak\n8. Cari ID Film\n9. Urut Data Rating Film-DESC\n10. Keluar");
18             System.out.println("=====");
19             pilih = sc.nextInt();

```

```

21     int id;
22     String judul;
23     double rating;
24     switch (pilih) {
25         case 1:
26             System.out.println("Masukkan Data Posisi Awal");
27             System.out.println("ID Film");
28             id = sc.nextInt();
29             sc.nextLine();
30             System.out.println("Judul Film");
31             judul = sc.nextLine();
32             System.out.println("Rating");
33             rating = sc.nextDouble();
34             sc.nextLine();
35             dll.addFirst(id, judul, rating);
36             break;
37         case 2:
38             System.out.println("Masukkan Data Posisi Awal");
39             System.out.println("ID Film");
40             id = sc.nextInt();
41             sc.nextLine();
42             System.out.println("Judul Film");
43             judul = sc.nextLine();
44             System.out.println("Rating");
45             rating = sc.nextDouble();
46             sc.nextLine();
47             dll.addLast(id, judul, rating);
48             break;
49         case 3:
50             System.out.println("Masukkan Film Urutan ke-");
51             System.out.println("ID Film");
52             id = sc.nextInt();
53             sc.nextLine();
54             System.out.println("Judul Film");
55             judul = sc.nextLine();
56             System.out.println("Rating");
57             rating = sc.nextDouble();
58             sc.nextLine();
59             System.out.println("Data film ini akan masuk diurutan ke- ");
60             int index = sc.nextInt();
61             sc.nextLine();
62             dll.add(index, id, judul, rating);
63             break;
64         case 4:
65             dll.removeFirst();
66             break;
67         case 5:
68             dll.removeLast();
69             break;
70         case 6:
71             System.out.println("Hapus Film Urutan ke-");
72             index = sc.nextInt();
73             sc.nextLine();
74             dll.remove(index);
75             break;
76         case 7:
77             dll.print();
78             break;
79         case 8:
80             System.out.println("Cari data");
81             System.out.println("Masukkan ID Film yang dicari");
82             id = sc.nextInt();
83             sc.nextLine();
84             dll.getId(id);
85             break;
86         case 9:
87             dll.sortRating();
88             dll.print();
89             break;
90         case 10:
91             break;
92     }
93 } while (pilih < 10);
94 }
95 }

```

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
```

```
=====
1
Masukkan Data Posisi Awal
ID Film
1567
Judul Film
the dark
Rating
8.4
```

```
=====
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
```

```
=====
3
Masukkan Film Urutan ke-
ID Film
1234
Judul Film
death
Rating
6.6
Data film ini akan masuk diurutkan ke-
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
```

```
=====
2
Masukkan Data Posisi Awal
ID Film
1346
Judul Film
uncharted
Rating
6.7
```

```
10. Keluar
=====
7
Cetak Data
ID: 1222
    Judul Film: spiderman
    Rating: 8.7
ID: 1765
    Judul Film: skyfall
    Rating: 7.8
ID: 1567
    Judul Film: the dark
    Rating: 8.4
ID: 1234
    Judul Film: death
    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 data
Masukkan ID Film yang dicari
1567
Data ID Film: 1567 berada di node ke-3
IDENTITAS:
  ID: 1567
  Judul Film: the dark
  Rating: 8.4

```

```

9. Urut Data Rating Film-DESC
10. Keluar
=====
9
Cetak Data
ID: 1222
  Judul Film: spiderman
  Rating: 8.7
ID: 1567
  Judul Film: the dark
  Rating: 8.4
ID: 1765
  Judul Film: skyfall
  Rating: 7.8
ID: 1346
  Judul Film: uncharted
  Rating: 6.7
ID: 1234
  Judul Film: death
  Rating: 6.6

```