Nama : Romy Mahardika Pangestu Lazuardi
No : 25
Kelas : 1H/D4TI

JOBSHEET IX

LINKED LIST

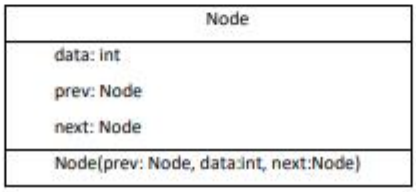
12.2 Kegiatan Praktikum 1

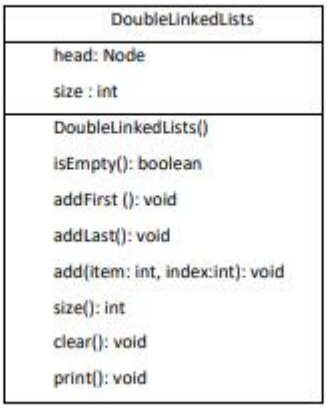
Waktu : 90 Menit

12.2.1 Percobaan 1

Pada percobaan 1 ini akan dibuat class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan linked list, belakang ataupun indeks tertentu pada linked list).

1. Perhatikan diagram class Node dan class DoublelinkedLists di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.





2. Buat paket baru dengan nama **doublelinkedlists**
3. Buat class di dalam paket tersebut dengan nama **Node**

```
package doublelinkedlists;

/**...4 lines */
public class Node {

}
```

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

```
4        int data;
5        Node prev, next;
```

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

```
7        Node(Node prev, int data, Node next){
8            this.prev=prev;
9            this.data=data;
10           this.next=next;
11       }
12   }
```

6. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:

```
package doublelinkedlists;

/**...4 lines */
public class DoubleLinkedLists {

}
```

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

```
8        Node head;
9        int size;
```

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

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

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

```
16       public boolean isEmpty(){
17           return head == null;
18       }
```

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

```
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++;
}
```

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

```
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++;
    }
}
```

12. 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("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++;
}
```

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

```
138     public int size(){
139         return size;
140     }
```

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

```
141     public void clear(){
142         head = null;
143         size = 0;
144     }
```

15. 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()) {
        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 Lists Kosong");
    }
}
```

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

```
package doublelinkedlists;

/**...4 lines */
public class DoubleLinkedListsMain {
    public static void main(String[] args) {

    }
}
```

17. Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

```
19          doubleLinkedList dll = new doubleLinkedList();
20          dll.print();
21          System.out.println("Size : "+dll.size());
22          System.out.println("=====================================");
23          dll.addFirst(3);
24          dll.addLast(4);
25          dll.addFirst(7);
26          dll.print();
27          System.out.println("Size : "+dll.size());
28          System.out.println("=====================================");
29          dll.add(40, 1);
30          dll.print();
31          System.out.println("Size : "+dll.size());
32          System.out.println("=====================================");
33          dll.clear();
34          dll.print();
35          System.out.println("Size : "+dll.size());
```

12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec
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
------------------------------------------------------------------------
```

```
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
```

```java
public class Node {
    int data;
    Node prev, next;

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

```java
public class DoubleLinkedLists {
    Node head;
    int size;
```

```
    public DoubleLinkedLists() {
        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);
                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 Lists Kosong");
        }
    }

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

```
public class DoubleLinkedListsMain {
public static void main(String[] args) {
    DoubleLinkedLists dll = new DoubleLinkedLists();
    dll.print();
    System.out.println("Size: " + dll.size());
    dll.addFirst(3);
```

```
    dll.addLast(4);
    dll.addFirst(7);
    dll.print();
    System.out.println("Size: " + dll.size());

    try {
        dll.add(40, 1);
        dll.print();
        System.out.println("Size: " + dll.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    dll.clear();
    dll.print();
    System.out.println("Size: " + dll.size());

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

    try {
        dll.removeFirst();
        dll.print();
        System.out.println("Size: " + dll.size());

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

        dll.remove(1);
        dll.print();
        System.out.println("Size: " + dll.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}
```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

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

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

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

---

5. Perhatikan pada method **addFirst().** Apakah arti statement head.prev = newNode ?
6. Perhatikan isi method **addLast()**, apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?

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

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

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

   jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

1. Perbedaan antara Single Linked List dengan Double Linked Lists:

Single Linked List: Setiap node hanya memiliki satu pointer yang menunjuk ke node berikutnya dalam urutan. Ini membuat traversal hanya bisa dilakukan satu arah, yaitu maju (dari head ke tail).

Double Linked List: Setiap node memiliki dua pointer: satu menunjuk ke node berikutnya (next) dan satu lagi menunjuk ke node sebelumnya (prev). Ini memungkinkan traversal dua arah, baik maju (dari head ke tail) maupun mundur (dari tail ke head).

2. Atribut next dan prev dalam class Node:

next: Pointer yang menunjuk ke node berikutnya dalam linked list.

prev: Pointer yang menunjuk ke node sebelumnya dalam linked list.

Kedua atribut ini memungkinkan traversal dalam dua arah pada double linked list.

3. Kegunaan inisialisasi atribut head dan size dalam konstruktor class DoubleLinkedLists:

head = null: Menginisialisasi bahwa linked list baru belum memiliki node, sehingga head diset ke null.

size = 0: Menginisialisasi ukuran linked list dengan nilai awal 0, karena belum ada node yang ditambahkan ke list.

4. Kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null dalam method addFirst():

Saat menambahkan node di awal list (addFirst), node baru tersebut akan menjadi node pertama. Karena tidak ada node sebelumnya (prev), maka prev diset ke null.

5. Arti dari statement head.prev = newNode dalam method addFirst():

Setelah node baru ditambahkan di awal list, node baru tersebut menjadi head yang baru, dan node yang sebelumnya menjadi head sekarang memiliki node sebelumnya (prev) yang menunjuk ke node baru. Ini menjaga keterkaitan dua arah pada double linked list.

6. Arti dari pembuatan object Node dengan parameter prev diset ke current, dan next diset ke null dalam method addLast():

Dalam method addLast, node baru akan ditambahkan di akhir list. Oleh karena itu:

prev = current: Node baru harus menunjuk ke node terakhir saat ini (yaitu current) sebagai node sebelumnya.

next = null: Node baru tidak memiliki node berikutnya, sehingga next diset ke null.

7. Penjelasan kode dalam method add():

while (i < index) { current = current.next; i++; }: Loop ini bergerak melalui list hingga mencapai posisi index yang diinginkan.

if (current.prev == null): Mengecek apakah node current adalah head (karena hanya head yang memiliki prev bernilai null).

Jika ya, buat node baru (newNode) yang menunjuk ke current sebagai node berikutnya dan null sebagai node sebelumnya.

Update prev dari current untuk menunjuk ke newNode.

Update head untuk menunjuk ke newNode karena node baru ini sekarang adalah head.

else: Jika current bukan head,

Buat node baru (newNode) yang menunjuk ke node sebelumnya (current.prev) dan current sebagai node berikutnya.

Update next dari node sebelumnya (current.prev) untuk menunjuk ke newNode.

Update prev dari current untuk menunjuk ke newNode.

12.3 Kegiatan Praktikum 2

Waktu : 60 Menit

12.3.1 Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists. Method tambahan tersebut akan ditambahkan sesuai pada diagram class berikut ini.

DoubleLinkedLists
head: Node size : int
DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void **removeFirst(): void** **removeLast(): void** **remove(index:int):void**

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

```
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--;
    }
}
```

2. Tambahkan method **removeLast()** di dalam class **DoubleLinkedLists.**

```
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--;
}
```

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

```
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--;
    }
}
```

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

```
42              dll.addLast(50);
43              dll.addLast(40);
44              dll.addLast(10);
45              dll.addLast(20);
46              dll.print();
47              System.out.println("Size : "+dll.size());
48              System.out.println("==================================");
49              dll.removeFirst();
50              dll.print();
51              System.out.println("Size : "+dll.size());
52              System.out.println("==================================");
53              dll.removeLast();
54              dll.print();
55              System.out.println("Size : "+dll.size());
56              System.out.println("==================================");
57              dll.remove(1);
58              dll.print();
59              System.out.println("Size : "+dll.size());
```

12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec
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
------------------------------
```

```java
public class DoubleLinkedLists {
    Node head;
    int size;

    public DoubleLinkedLists() {
        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);
                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 Lists Kosong");
        }
    }

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

```
public class DoubleLinkedListsMain {
public static void main(String[] args) {
    DoubleLinkedLists dll = new DoubleLinkedLists();
    dll.print();
    System.out.println("Size: " + dll.size());
    dll.addFirst(3);
    dll.addLast(4);
    dll.addFirst(7);
    dll.print();
    System.out.println("Size: " + dll.size());

    try {
        dll.add(40, 1);
        dll.print();
        System.out.println("Size: " + dll.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }

    dll.clear();
    dll.print();
    System.out.println("Size: " + dll.size());
```

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

    try {
        dll.removeFirst();
        dll.print();
        System.out.println("Size: " + dll.size());

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

        dll.remove(1);
        dll.print();
        System.out.println("Size: " + dll.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}
```

```
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
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;
   ```

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method **removeLast()**?
3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah **remove**!

   ```
   Node tmp = head.next;

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

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

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

1. Maksud statement berikut pada method removeFirst():
   - head = head.next;: Statement ini memindahkan head ke node berikutnya. Artinya, node yang awalnya menjadi head akan dihapus, dan node kedua dalam list sekarang menjadi head baru.
   - head.prev = null;: Setelah memperbarui head, statement ini mengatur atribut prev dari node head yang baru ke null karena node head tidak memiliki node sebelumnya. Ini menjaga integritas double linked list dengan memastikan bahwa head baru tidak memiliki referensi ke node yang telah dihapus.

2. Cara mendeteksi posisi data ada pada bagian akhir pada method removeLast():

   Untuk mendeteksi posisi data pada bagian akhir dalam method removeLast(), langkah yang umum dilakukan adalah menelusuri list hingga menemukan node terakhir.
   Loop while (current.next != null) akan berhenti ketika current.next bernilai null, yang berarti current adalah node terakhir dalam list. Node terakhir ini kemudian dapat dihapus dengan memperbarui referensi next dari node sebelumnya (current.prev.next) menjadi null.

3. Alasan potongan kode program berikut tidak cocok untuk perintah remove:

Potongan kode ini tidak cocok untuk operasi remove karena tidak menghapus node pertama dengan benar dan tidak memperhitungkan semua skenario penghapusan node dalam list. Berikut alasannya:

- Node tmp = head.next;: Mengatur tmp untuk menunjuk ke node kedua dalam list.
- head.next = tmp.next;: Memperbarui next dari head sehingga melompati node tmp dan menunjuk ke node setelah tmp.
- tmp.next.prev = head;: Memperbarui prev dari node setelah tmp sehingga menunjuk kembali ke head.

Alasan kode ini tidak cocok:

- Kode ini tidak menghapus node tmp dari memori (meskipun referensi hilang, pengelolaan memori yang baik tetap perlu dipertimbangkan).
- Tidak ada penanganan untuk kasus list dengan satu elemen saja.
- Tidak mempertimbangkan jika tmp.next adalah null (yaitu, node kedua adalah node terakhir).

4. Fungsi kode program berikut ini pada fungsi remove:

Kode ini digunakan untuk menghapus node current dari double linked list dengan benar. Berikut adalah penjelasan detailnya:

- current.prev.next = current.next;: Mengatur referensi next dari node sebelumnya (current.prev) untuk melompati current dan menunjuk ke node setelah current (current.next). Ini menghapus referensi ke current dari node sebelumnya.
- current.next.prev = current.prev;: Mengatur referensi prev dari node berikutnya (current.next) untuk melompati current dan menunjuk ke node sebelum current (current.prev). Ini menghapus referensi ke current dari node berikutnya.

Dengan melakukan kedua operasi ini, node current sepenuhnya terputus dari linked list, memungkinkan pengelolaan memori untuk node current dilakukan oleh garbage collector jika menggunakan Java atau menghapus node secara manual dalam bahasa lain. Kode ini memastikan bahwa linked list tetap terhubung dengan benar setelah penghapusan node.

12.4 Kegiatan Praktikum 3

Waktu : 50 Menit

12.4.1 Tahapan Percobaan

Pada praktikum 3 ini dilakukan uji coba untuk mengambil data pada linked list dalam 3 kondisi, yaitu mengambil data paling awal, paling akhir dan data pada indeks tertentu dalam linked list. Method mengambil data dinamakan dengan **get**. Ada 3 method get yang dibuat pada praktikum ini sesuai dengan diagram class DoubleLinkedLists.

DoubleLinkedLists
head: Node size : int
DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void removeFirst(): void removeLast(): void remove(index:int):void **getFirst(): int** **getLast() : int** **get(index:int): int**

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("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("Linked List kosong");
    }
    Node tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

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

```
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;
}
```

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

```
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));
```

12.4.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec (default-cli)
Linked Lists Kosong
Size: 0
========================
7       3       4
berhasil diisi
Size: 3
========================
7       40      3       4
berhasil diisi
Size: 4
========================
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

```
public class DoubleLinkedListsMain {
    public static void main(String[] args) {
        DoubleLinkedLists dll = new DoubleLinkedLists();

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

```
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size: " + dll.size());

        try {
            dll.add(40, 1);
            dll.print();
            System.out.println("Size: " + dll.size());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        try {
            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));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
Linked Lists kosong!
Size: 0
7       3       4
>Berhasil diisi!
Size: 3
7       40      3       4
>Berhasil diisi!
Size: 4
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method **size()** pada class DoubleLinkedLists!
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!
3. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

```
public boolean isEmpty(){
    if(size ==0){
        return true;
    } else{
        return false;
    }
}
```

(a)

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

(b)

1. Jelaskan method size() pada class DoubleLinkedLists!

Method size() pada class DoubleLinkedLists berfungsi untuk mengembalikan jumlah elemen atau node yang ada dalam linked list. Method ini sederhana karena hanya mengembalikan nilai dari atribut size yang terus diperbarui setiap kali ada penambahan atau penghapusan node dalam linked list.

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

Secara konvensional, indeks dalam struktur data dimulai dari 0. Namun, jika ingin mengatur indeks supaya dimulai dari 1, Anda harus melakukan penyesuaian dalam semua operasi yang melibatkan indeks.

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

Karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists memiliki perbedaan utama sebagai berikut:

- Double Linked Lists:
    - Setiap node memiliki dua pointer: prev yang menunjuk ke node sebelumnya dan next yang menunjuk ke node berikutnya.
    - Saat menambahkan node, Anda harus memperbarui next dari node sebelumnya dan prev dari node berikutnya (jika ada).
    - Menambahkan node di tengah list memerlukan penyesuaian dua arah, baik prev maupun next, untuk menjaga keterkaitan dua arah.

Single Linked Lists:

- Setiap node hanya memiliki satu pointer: next yang menunjuk ke node berikutnya.
- Saat menambahkan node, Anda hanya perlu memperbarui next dari node sebelumnya.
- Menambahkan node di tengah list hanya memerlukan penyesuaian satu arah, yaitu next.

Karena double linked lists memiliki pointer dua arah (prev dan next), manipulasi mereka biasanya lebih kompleks tetapi memungkinkan traversal dua arah.

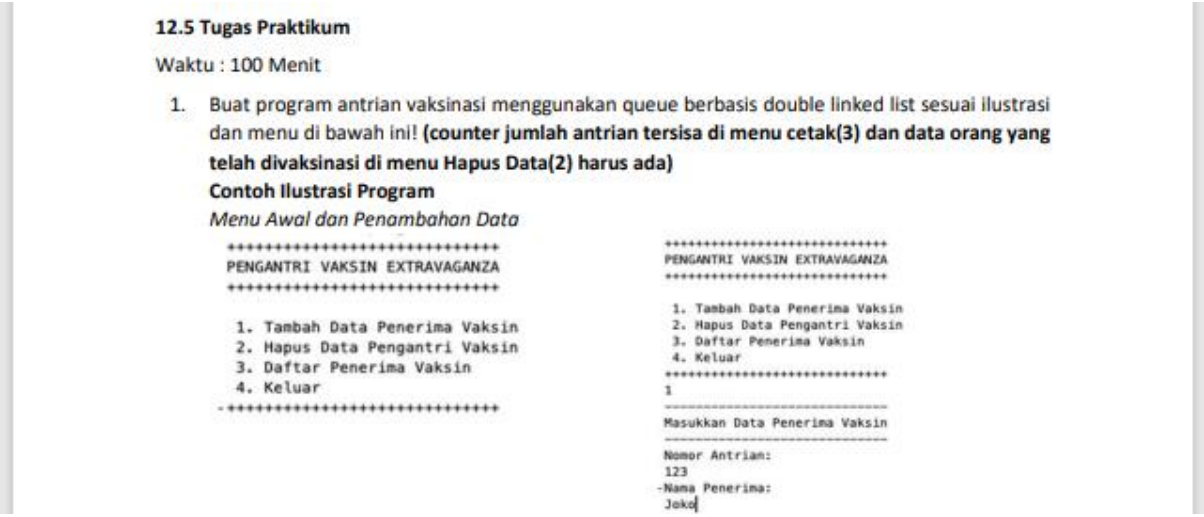
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

- (a): Method ini menentukan apakah linked list kosong berdasarkan nilai atribut size.
    - Jika size adalah 0, method mengembalikan true, menunjukkan bahwa linked list kosong.
    - Jika size bukan 0, method mengembalikan false, menunjukkan bahwa linked list tidak kosong.
    - Ini mengandalkan penghitung ukuran (size) untuk memeriksa kekosongan.
- (b): Method ini menentukan apakah linked list kosong berdasarkan apakah head adalah null.
    - Jika head adalah null, method mengembalikan true, menunjukkan bahwa linked list kosong.
    - Jika head bukan null, method mengembalikan false, menunjukkan bahwa linked list tidak kosong.
    - Ini mengandalkan pointer head untuk memeriksa kekosongan.

Perbedaan Utama:

- (a): Menggunakan atribut size untuk menentukan kekosongan, lebih eksplisit dalam mengikuti perubahan ukuran list.
- (b): Menggunakan pointer head untuk menentukan kekosongan, lebih langsung memeriksa apakah ada elemen dalam list tanpa memperhatikan ukuran.

Kedua pendekatan valid, tetapi (a) lebih cocok jika ukuran list terus dikelola dengan benar, sedangkan (b) lebih langsung tetapi bergantung pada head yang diperbarui secara tepat.



12.5 Tugas Praktikum

Waktu : 100 Menit

1. Buat program antrian vaksinasi menggunakan queue berbasis double linked list sesuai ilustrasi dan menu di bawah ini! **(counter jumlah antrian tersisa di menu cetak(3) dan data orang yang telah divaksinasi di menu Hapus Data(2) harus ada)**

Contoh Ilustrasi Program

Menu Awal dan Penambahan Data

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

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

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

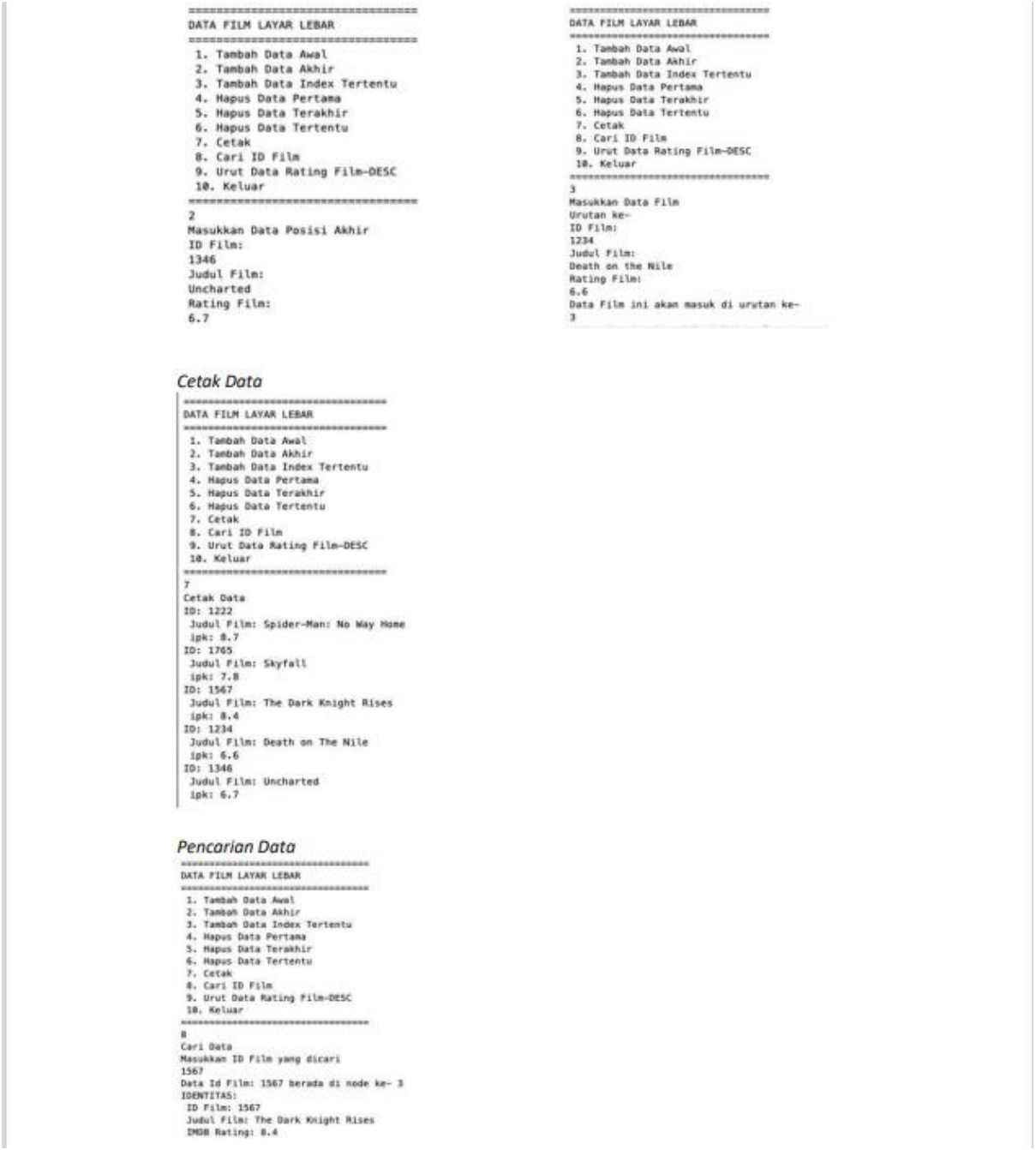
2. Buatlah program daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists, bentuk program memiliki fitur pencarian melalui ID Film dan pengurutan Rating secara descending. Class Film wajib diimplementasikan dalam soal ini.

Contoh Ilustrasi Program

Menu Awal dan Penambahan Data

```
==================================
DATA FILM LAYAR LEBAR
==================================
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film-DESC
 10. Keluar
==================================
```

```
==================================
DATA FILM LAYAR LEBAR
==================================
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film-DESC
 10. Keluar
==================================
1
Masukkan Data Film Posisi Awal
ID Film:
1222
Judul Film:
Spider-Man: No Way Home
Rating Film:
8.7
```



Tugas

1. Vaksin

```
public class Node {
    int nomorAntrian;
    String nama;
    Node prev, next;

    public Node(Node prev, int nomorAntrian, String nama, Node next) {
        this.prev = prev;
        this.nomorAntrian = nomorAntrian;
        this.nama = nama;
        this.next = next;
```

```
    }
}
```

```
public class DoubleLinkedListQueue {
    Node head, tail;
    int size;

    public DoubleLinkedListQueue() {
        head = null;
        tail = null;
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public void enqueue(int nomorAntrian, String nama) {
        Node newNode = new Node(null, nomorAntrian, nama, null);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
    }

    public void dequeue() throws Exception {
        if (isEmpty()) {
            throw new Exception("Antrian masih kosong!");
        } else {
            System.out.println(head.nama + " telah selesai divaksinasi.");
            head = head.next;
            if (head != null) {
                head.prev = null;
            } else {
                tail = null;
            }
            size--;
        }
    }

    public void printQueue() {
        if (!isEmpty()) {
            Node current = head;
```

```
            System.out.println("Daftar Pengantri Vaksin:");
            while (current != null) {
                System.out.println("No. " + current.nomorAntrian + " Nama: " +
current.nama);
                current = current.next;
            }
            System.out.println("Sisa Antrian: " + size);
        } else {
            System.out.println("Antrian kosong.");
        }
    }
}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoubleLinkedListQueue queue = new DoubleLinkedListQueue();
        int choice;
        do {
            System.out.println("PENGANTRI VAKSIN EXTRAVAGANZA");
            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.print("Pilih menu: ");
            choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Masukkan Nomor Antrian: ");
                    int nomorAntrian = scanner.nextInt();
                    scanner.nextLine();  // Consume newline
                    System.out.print("Masukkan Nama Penerima: ");
                    String nama = scanner.nextLine();
                    queue.enqueue(nomorAntrian, nama);
                    break;
                case 2:
                    try {
                        queue.dequeue();
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                case 3:
```

```
                    queue.printQueue();
                    break;
                case 4:
                    System.out.println("Terima kasih!");
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
                    break;
            }
        } while (choice != 4);
        scanner.close();
    }
}
```

```
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilih menu: 1
Masukkan Nomor Antrian: 123
Masukkan Nama Penerima: Joko
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilih menu: 3
Daftar Pengantri Vaksin:
No. 123 Nama: Joko
Sisa Antrian: 1
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilih menu: 2
Joko telah selesai divaksinasi.
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
```

2. Film

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

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

```
•       }
•   }
•
```

```
public class DoubleLinkedListFilm {
    Film head, tail;
    int size;

    public DoubleLinkedListFilm() {
        head = null;
        tail = null;
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public void addFirst(int id, String judul, double rating) {
        Film newNode = new Film(null, id, judul, rating, head);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void addLast(int id, String judul, double rating) {
        Film newNode = new Film(tail, id, judul, rating, null);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
    }

    public void add(int id, String judul, double rating, int index) throws
Exception {
        if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        if (index == 0) {
            addFirst(id, judul, rating);
```

```
        } else if (index == size) {
            addLast(id, judul, rating);
        } else {
            Film current = head;
            for (int i = 0; i < index; i++) {
                current = current.next;
            }
            Film newNode = new Film(current.prev, id, judul, rating, current);
            current.prev.next = newNode;
            current.prev = newNode;
            size++;
        }
    }

    public void removeFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Daftar film kosong.");
        } else {
            head = head.next;
            if (head != null) {
                head.prev = null;
            } else {
                tail = null;
            }
            size--;
        }
    }

    public void removeLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Daftar film kosong.");
        } else {
            tail = tail.prev;
            if (tail != null) {
                tail.next = null;
            } else {
                head = null;
            }
            size--;
        }
    }

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

```
        } else if (index == size - 1) {
            removeLast();
        } else {
            Film current = head;
            for (int i = 0; i < index; i++) {
                current = current.next;
            }
            current.prev.next = current.next;
            current.next.prev = current.prev;
            size--;
        }
    }

    public Film searchById(int id) {
        Film current = head;
        while (current != null) {
            if (current.id == id) {
                return current;
            }
            current = current.next;
        }
        return null;
    }

    public void sortByRatingDesc() {
        if (isEmpty() || size == 1) {
            return;
        }
        boolean swapped;
        do {
            swapped = false;
            Film current = head;
            while (current.next != null) {
                if (current.rating < current.next.rating) {
                    // Swap data
                    int tempId = current.id;
                    String tempJudul = current.judul;
                    double tempRating = current.rating;

                    current.id = current.next.id;
                    current.judul = current.next.judul;
                    current.rating = current.next.rating;

                    current.next.id = tempId;
                    current.next.judul = tempJudul;
                    current.next.rating = tempRating;

                    swapped = true;
```

```
                }
                current = current.next;
            }
        } while (swapped);
    }

    public void printFilms() {
        if (!isEmpty()) {
            Film current = head;
            while (current != null) {
                System.out.println("ID: " + current.id + ", Judul: " +
current.judul + ", Rating: " + current.rating);
                current = current.next;
            }
        } else {
            System.out.println("Daftar film kosong.");
        }
    }
}
```

```
import java.util.Scanner;

public class MainFilm {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoubleLinkedListFilm filmList = new DoubleLinkedListFilm();
        int choice;
        do {
            System.out.println("DATA FILM LAYAR LEBAR");
            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.print("Pilih menu: ");
            choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("ID Film: ");
                    int id = scanner.nextInt();
```

```
                scanner.nextLine();  // Consume newline
                System.out.print("Judul Film: ");
                String judul = scanner.nextLine();
                System.out.print("Rating Film: ");
                double rating = scanner.nextDouble();
                filmList.addFirst(id, judul, rating);
                break;
            case 2:
                System.out.print("ID Film: ");
                id = scanner.nextInt();
                scanner.nextLine();  // Consume newline
                System.out.print("Judul Film: ");
                judul = scanner.nextLine();
                System.out.print("Rating Film: ");
                rating = scanner.nextDouble();
                filmList.addLast(id, judul, rating);
                break;
            case 3:
                System.out.print("ID Film: ");
                id = scanner.nextInt();
                scanner.nextLine();  // Consume newline
                System.out.print("Judul Film: ");
                judul = scanner.nextLine();
                System.out.print("Rating Film: ");
                rating = scanner.nextDouble();
                System.out.print("Urutan ke-: ");
                int index = scanner.nextInt();
                try {
                    filmList.add(id, judul, rating, index);
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
                break;
            case 4:
                try {
                    filmList.removeFirst();
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
                break;
            case 5:
                try {
                    filmList.removeLast();
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
                break;
            case 6:
```

```
                    System.out.print("Masukkan indeks yang ingin dihapus: ");
                    index = scanner.nextInt();
                    try {
                        filmList.remove(index);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                case 7:
                    filmList.printFilms();
                    break;
                case 8:
                    System.out.print("Masukkan ID Film yang dicari: ");
                    id = scanner.nextInt();
                    Film foundFilm = filmList.searchById(id);
                    if (foundFilm != null) {
                        System.out.println("ID Film: " + foundFilm.id);
                        System.out.println("Judul Film: " + foundFilm.judul);
                        System.out.println("IMDB Rating: " +
foundFilm.rating);
                    } else {
                        System.out.println("Film dengan ID " + id + " tidak
ditemukan.");
                    }
                    break;
                case 9:
                    filmList.sortByRatingDesc();
                    System.out.println("Data telah diurutkan.");
                    break;
                case 10:
                    System.out.println("Terima kasih!");
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
                    break;
            }
        } while (choice != 10);
        scanner.close();
    }
}
```

```
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: 1
ID Film: 1222
Judul Film: Spiderman
Rating Film: 8,7
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
Pilih menu: 2
ID Film: 1346
Judul Film: Uncha
Rating Film: 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
Pilih menu: 3
ID Film: 1234
Judul Film: DON
Rating Film: 6,6
Urutan ke-: 3
Nilai indeks di luar batas
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
ID: 1222, Judul: Spiderman, Rating: 8.7
ID: 1346, Judul: Uncha, Rating: 6.7
```

```
ID: 1222, Judul: Spiderman, Rating: 8.7
ID: 1346, Judul: Uncha, 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
Pilih menu: 8
Masukkan ID Film yang dicari: 1234
Film dengan ID 1234 tidak ditemukan.
```