LAPORAN PRAKTIKUM

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

PERTEMUAN 12 : DOUBLE LINKED LISTS

KAYLA RACHMAUDINA SATITI PUTRI

2341760103

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI POLITEKNIK

NEGERI MALANG

2024

JOBSHEET 10
Double Linked Lists

12.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami algoritma double linked lists;
2. membuat dan mendeklarasikan struktur algoritma double linked lists;
3. menerapkan algoritma double linked lists dalam beberapa *study case*.

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.

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

DoubleLinkedLists
head: Node size : int

DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void

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.

```
1  public class Node15 {
2      int data;
3      Node15 prev, next;
```

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

```
5      Node15(Node15 prev, int data, Node15 next) {
6          this.data = data;
7          this.next = next;
8          this.prev = prev;
9      }
```

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.

```
1  public class DoubleLinkedLists15 {
2      Node15 head;
3      int size;
```

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

```
5   public DoubleLinkedLists15(){
6       head = null;
7       size = 0;
8   }
```

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

```
10  public boolean isEmpty() {
11      return head == null;
12  }
```

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

```
14  public void addFirst(int item) {
15      if (isEmpty()) {
16          head = new Node15(prev:null, item, next:null);
17      } else {
18          Node15 newNode = new Node15(prev:null, item, head);
19          head.prev = newNode;
20          head = newNode;
21      }
22      size++;
23  }
```

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

```
25  public void addLast(int item) {
26      if (isEmpty()) {
27          addFirst(item);
28      } else {
29          Node15 current = head;
30          while (current.next != null) {
31              current = current.next;
32          }
33          Node15 newNode = new Node15(current, item, next:null);
34          current.next = newNode;
35          size++;
36      }
37  }
```

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(message:"Nilai index di luar batas");
    } else{
        Node15 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            Node15 newNode = new Node15(prev:null, item, current);
            newNode.prev = newNode;
            head = newNode;
        } else {
            Node15 newNode = new Node15(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.

```
69      public void clear() {
70          head = null;
71          size = 0;
72      }
```

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

```
69      public void clear() {
70          head = null;
71          size = 0;
72      }
```

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()) {
        Node15 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");
    }
}
```

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

```
1   public class DoubleLinkedListsMain15 {
        Run | Debug
2       public static void main(String[] args) throws Exception {
3           System.out.println();
```

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

```
        DoubleLinkedLists15 dll = new DoubleLinkedLists15();
```

```
        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:"============================");
        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());
    }
```

12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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
```

12.2.3 Pertanyaan Percobaan

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

   Single linked list hanya memungkinkan navigasi ke depan, sedangkan double linked list memungkinkan navigasi ke depan dan ke belakang.

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

   Atribut next adalah pointer yang menunjuk ke node berikutnya dalam linked list. Sedangkan atribut prev adalah pointer yang menunjuk ke node sebelumnya dalam linked list.

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

   Head nya null untuk menandakan daftar kosong. Size nya 0 untuk menandakan kalau tidk ada node.

```
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);

   Untuk menunjukkan bahwa node baru tersebut adalah head baru dari linked list, dan tidak ada node sebelumnya yang harus dihubungkan ke node baru ini.

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

   Untuk mengatur pointer prev pada node head saat ini ke node yang baru ditmbahkan (newNode) dalam method addFirst pada double linked list.

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**);

Dengan mengatur prev sebagai current, kita menghubungkan node baru dengan node terakhir dalam linked list, sehingga membuatnya menjadi node berikutnya dari node terakhir tersebut. Dan dengan mengatur next sebagai null, kita menunjukkan bahwa node baru tersebut adalah node terakhir dalam linked list, karena tidak ada node setelahnya.

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.

Memeriksa apakah atribut prev dari node current bernilai null atau tidak. Jika iya, maka dibuatlah node baru (newNode) dengan item yang diberikan dan atribut prevnya diatur untuk merujuk ke node itu sendiri (newNode.prev = 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.**

```java
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.**

```java
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;
    }
    Node15 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.

```
29  public void remove(int index) throws Exception {
30      if (isEmpty() || index >= size) {
31          throw new Exception("Nilai index di luar batas");
32      } else if (index == 0) {
33          removeFirst();
34      } else {
35          Node15 current = head;
36          int i = 0;
37          while (i < index) {
38              current = current.next;
39              i++;
40          }
41          if (current.next == null) {
42              current.prev.next = null;
43          } else if (current.prev == null) {
44              current = current.next;
45              current.prev = null;
46              head = current;
47          } else {
48              current.prev.next = current.next;
49              current.next.prev = current.prev;
50          }
51          size--;
52      }
53  }
```

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

```
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());
```

12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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;

   Digunakan untuk menghapus node pertama pada linked list.

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

   Dengan perulangan while yang bekerja mencari node sebelum node terahir.

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;
```

   Kode tersebut hanya mengubah pointer next dari node head dan tmp, tidak mengubah struktur data yang mendasarinya. Node yang ingin dihapus masih ada di dalam memori, meskipun tidak dapat diakses lagi melalui daftar berantai.

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

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

   Digunakan untuk mengupdate rantai node-node pada linked list setelah node yang akan dihapus.

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.

```
141     public int getFirst() throws Exception {
142         if (isEmpty()) {
143             throw new Exception(message:"Linked list kosong");
144         }
145         return head.data;
146     }
```

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

```
148     public int getLast() throws Exception {
149         if (isEmpty()) {
150             throw new Exception(message:"Linked list kosong");
151         }
152         Node15 tmp = head;
153         while (tmp.next != null) {
154             tmp = tmp.next;
155         }
156         return tmp.data;
157     }
```

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

```
159     public int get(int index) throws Exception {
160         if (isEmpty() || index >= size) {
161             throw new Exception(message:"Nilai index di luar batas");
162         }
163         Node15 tmp = head;
164         for (int i = 0; i < index; i++) {
165             tmp = tmp.next;
166         }
167         return tmp.data;
168     }
```

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

```
1       dll.print();
2       System.out.println("Size : " + dll.size());
3       System.out.println("============================");
4       dll.addFirst(3);
5       dll.addLast(4);
6       dll.addFirst(7);
7       dll.print();
8       System.out.println("Size : " + dll.size());
9       System.out.println("============================");
10      dll.add(40, 1);
11      dll.print();
12      System.out.println("Size : " + dll.size());
13      System.out.println("================================================");
14      System.out.println("Data awal pada Linked Lists adalah        : " + dll.getFirst());
15      System.out.println("Data akhir pada Linked Lists adalah       : " + dll.getLast());
16      System.out.println("Data indeks ke-1 pada Linked Lists adalah : " + dll.get(1));
17      System.out.println("================================================");
18  }
```

12.4.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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!

   variabel size adalah variabel instan yang mengikuti jumlah node dalam linked list. Nilainya diinisialisasi ke 0 di konstruktor dan diincrement setiap kali node baru ditambahkan ke dalam linked list.

   Metode size() hanya menghasilkan nilai saat ini dari variabel size, menyediakan cara yang cepat dan mudah untuk menentukan panjang linked list saat ini.

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

   Dengan menambahkan atribut index ke kelas Node dan memperbarui metode insert(), get(), dan remove() untuk memperhitungkan indeks baru ini. Modifikasi ini memungkinkan kita untuk mengakses elemen dengan cara yang lebih intuitif dan konsisten dengan struktur data lain seperti array.

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

   Fungsi add pada Double Linked Lists dan Single Linked Lists digunakan untuk menambahkan elemen baru ke akhir daftar. Namun, proses penambahan elemen ini berbeda-beda antara Double Linked Lists dan Single Linked Lists.

   Pada Single Linked List, fungsi add hanya membutuhkan referensi ke node akhir saat ini dan membuat node baru dengan nilai yang diinginkan. Kemudian, node baru ini diset sebagai node berikutnya dari node akhir saat ini. Jika node akhir saat ini kosong, maka node baru menjadi node pertama dan node utama.

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

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

(a)

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

(b)

Kode (a):

Menggunakan operator == untuk membandingkan nilai size dengan 0. Jika size sama dengan 0, maka list kosong. Kemudian mengembalikan nilai true jika size nya kosong.

Kode (b):

Langsung mengecek nilai head. Jika head nya bernilai null, maka list kosong kemudian langsung mengembalikan nilai true.

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)**

   Class NodeVaksin15

```java
public class NodeVaksin15 {
    Vaksinasi15 data;
    NodeVaksin15 prev, next;

    NodeVaksin15(NodeVaksin15 prev, Vaksinasi15 data, NodeVaksin15 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
```

   Class Vaksinasi15

```java
public class Vaksinasi15 {
    String nama;
    int noAntri;

    Vaksinasi15() {}

    Vaksinasi15(int noAntri, String nama) {
        this.noAntri = noAntri;
        this.nama = nama;
    }
```

Class VaksinasiDLL15

```java
public class VaksinasiDLL15 {
    NodeVaksin15 head;
    int size;

    public VaksinasiDLL15(){
        head = null;
        size = 0;
    }

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

    public void addFirst(Vaksinasi15 item) {
        if (isEmpty()) {
            head = new NodeVaksin15(null, item, null);
        } else {
            NodeVaksin15 newNode = new NodeVaksin15(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void addLast(Vaksinasi15 item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            NodeVaksin15 current = head;
            while (current.next != null) {
                current = current.next;
            }
            NodeVaksin15 newNode = new NodeVaksin15(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    public void add(Vaksinasi15 item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai index di luar batas");
        } else{
            NodeVaksin15 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                NodeVaksin15 newNode = new NodeVaksin15(null, item, current);
                newNode.prev = newNode;
                head = newNode;
            } else {
                NodeVaksin15 newNode = new NodeVaksin15(current.prev, item, current);
                newNode.prev = current.prev;
                newNode.next = current;
                current.prev.next = newNode;
                current.prev = newNode;
            }
        } size++;
    }
```

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

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

public void print() {
    if (!isEmpty()) {
        NodeVaksin15 tmp = head;
        while (tmp != null) {
            System.out.print(tmp.data.noAntri + " - " + tmp.data.nama + "\n");
            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;
    }
    NodeVaksin15 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 index di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
        NodeVaksin15 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--;
    }
}
```

```java
public Vaksinasi15 getFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked list kosong");
    }
    return head.data;
}

public Vaksinasi15 getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked list kosong");
    }
    NodeVaksin15 tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }
    return tmp.data;
}

public Vaksinasi15 get(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai index di luar batas");
    }
    NodeVaksin15 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

Main class

```java
import java.util.Scanner;

public class MainVaksin15 {
    public static void menu() {
        System.out.println("++++++++++++++++++++++++++++++");
        System.out.println("PENGANTRI VAKSIN EXTRAVAGANZA");
        System.out.println("++++++++++++++++++++++++++++++");
        System.out.println();
        System.out.println("1. Tambah Data Penerima Vaksin");
        System.out.println("2. Hapus Data Penerima Vaksin");
        System.out.println("3. Daftar Penerima Vaksin");
        System.out.println("4. Keluar");
        System.out.println("++++++++++++++++++++++++++++++");
    }
    public static void main(String[] args) throws Exception {
        VaksinasiDLL15 dll = new VaksinasiDLL15();
        Scanner sc = new Scanner(System.in);
        int pilih;
        do {
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    System.out.println("----------------------------");
                    System.out.println("Masukkan Data Penerima Vaksin");
                    System.out.println("----------------------------");
                    System.out.print("Nomor antrian : ");
                    int noAntrian = sc.nextInt();
                    System.out.print("Nama Penerima : ");
                    String nama = sc.next();
                    Vaksinasi15 nb = new Vaksinasi15(noAntrian, nama);
                    dll.addLast(nb);
                    System.out.println();
                    break;
                    case 2:
                    Vaksinasi15 penerima = dll.getFirst();
                    System.out.println(penerima.nama + " telah divaksinasi.");
                    dll.removeFirst();
                    break;
                    case 3:
                        System.out.println("----------------------------");
                        System.out.println("Daftar Pengantri Vaksin");
                        System.out.println("----------------------------");
                        dll.print();
                        System.out.println("Sisa Antrian : " + dll.size());
                        System.out.println();
                        break;
                    case 4:
                        return;
            }
        } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4);
    }
}
```

Output

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

1. Tambah Data Penerima Vaksin
2. Hapus Data Penerima Vaksin
3. Daftar Penerima Vaksin
4. Keluar
++++++++++++++++++++++++++++
1
----------------------------
Masukkan Data Penerima Vaksin
----------------------------
Nomor antrian : 1
Nama Penerima : ani

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

1. Tambah Data Penerima Vaksin
2. Hapus Data Penerima Vaksin
3. Daftar Penerima Vaksin
4. Keluar
++++++++++++++++++++++++++++
1
----------------------------
Masukkan Data Penerima Vaksin
----------------------------
Nomor antrian : 2
Nama Penerima : budi
```

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

1. Tambah Data Penerima Vaksin
2. Hapus Data Penerima Vaksin
3. Daftar Penerima Vaksin
4. Keluar
++++++++++++++++++++++++++++
3
----------------------------
Daftar Pengantri Vaksin
----------------------------
1 - ani
2 - budi
3 - caca

berhasil diisi
Sisa Antrian : 3
```

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

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

1. Tambah Data Penerima Vaksin
2. Hapus Data Penerima Vaksin
3. Daftar Penerima Vaksin
4. Keluar
++++++++++++++++++++++++++++
2
ani telah divaksinasi.
++++++++++++++++++++++++++++
PENGANTRI VAKSIN EXTRAVAGANZA
++++++++++++++++++++++++++++

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

berhasil diisi
Sisa Antrian : 2
```

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.

   Class Film15

```java
public class Film15 {
    int id;
    String judul;
    double rating;

    Film15() {}

    Film15(int id, String judul, double rating) {
        this.id = id;
        this.judul = judul;
        this.rating = rating;
    }
```

   Class DoubleLinkedLists15, isinya sama dengan bberapa tambahan

```java
    public Film15 searchById(int id) throws Exception {
        if (isEmpty()) {
            throw new Exception(message:"Linked List masih kosong");
        }

        Node15 current = head;
        while (current != null) {
            if (current.data.id == id) {
                return current.data;
            }
            current = current.next;
        }

        throw new Exception("Film dengan ID " + id + " tidak ditemukan");
    }
```

```java
public void sortByRatingDesc() {
    if (isEmpty() || size == 1) {
        return;
    }

    for (int i = 0; i < size - 1; i++) {
        Node15 current = head;
        Node15 maxNode = current;

        for (int j = 0; j < size - i - 1; j++) {
            if (current.next != null && current.next.data.rating > maxNode.data.rating) {
                maxNode = current.next;
            }
            current = current.next;
        }

        if (maxNode != current) {
            Film15 temp = current.data;
            current.data = maxNode.data;
            maxNode.data = temp;
        }
    }
```

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              : 123
Judul Film      : red
Rating (ex. 4.5) : 4.5
```

```
=====================================
2
---------------------------------
Masukkan Data Posisi Akhir
---------------------------------
ID              : 124
Judul Film      : blue
Rating (ex. 4.5) : 4.3
```

```
3
---------------------------------
Masukkan Data FILM
---------------------------------
Urutan ke       : 1
ID              : 125
Judul Film      : black
Rating (ex. 4.5) : 3.5
```

```
7
-----------------------------------
DATA FILM LAYAR LEBAR
-----------------------------------
Cetak Data
ID     : 124
Judul  : blue
Rating : 4.3
ID     : 125
Judul  : black
Rating : 3.5
ID     : 123
Judul  : red
Rating : 4.5
```

```
9
-----------------------------------
DATA FILM LAYAR LEBAR SORTING DESC
-----------------------------------
Cetak Data
ID     : 125
Judul  : black
Rating : 3.5
ID     : 124
Judul  : blue
Rating : 4.3
ID     : 123
Judul  : red
Rating : 4.5
```

```
=====================================
4
Film black telah dihapus.

=====================================
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
=====================================
5
Film red telah dihapus.
```

```
7
-----------------------------------
DATA FILM LAYAR LEBAR
-----------------------------------
Cetak Data
ID     : 124
Judul  : blue
Rating : 4.3
```