LAPORAN PRAKTIKUM JOBSHEET 8

DOUBLE LINKED LISTS

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

Disusun Oleh :

Jami'atul Afifah (2341760102)

SIB-1F

PROGRAM STUDI D4 SISTEM INFOEMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

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 Node14 dan class DoublelinkedLists14 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

```
doublelinkedlists14 > J Node.java > ...
1   package doublelinkedlists14;
2
3   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.

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

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

```
package doublelinkedlists14;

public class DoubleLinkedLists {
```

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

```
public class DoubleLinkedLists {
    Node head;
    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.

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

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(prev:null, item, next:null);
    } else {
        Node newNode = new Node(prev: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, next: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 (index < 0 || index > size) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        addFirst(item);
    } else if (index == size) {
        addLast(item);
    } else {
        Node current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        Node newNode = new Node(current.prev, item, 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.

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

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

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

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(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.

```
package doublelinkedlists14;

public class DoubleLinkedListsMain {
    Run | Debug
    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

```
package doublelinkedlists14;

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

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

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

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

Hasil Output :

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

PERTANYAAN PERCOBAAN

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
    - Single Linked List (SLL):
        - Setiap node hanya memiliki satu referensi, yaitu ke node berikutnya (next).
        - Traversal hanya bisa dilakukan ke satu arah, yaitu dari head ke tail.
        - Struktur lebih sederhana dan memerlukan memori lebih sedikit karena hanya menyimpan satu referensi per node.
    - Double Linked List (DLL):
        - Setiap node memiliki dua referensi, yaitu ke node sebelumnya (prev) dan ke node berikutnya (next).
        - Traversal bisa dilakukan ke dua arah, yaitu dari head ke tail dan sebaliknya.
        - Struktur lebih kompleks dan memerlukan memori lebih banyak karena menyimpan dua referensi per node.
        - Lebih fleksibel dalam operasi seperti penghapusan dan penambahan node di tengah.
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
    - next: Merupakan referensi ke node berikutnya dalam linked list. Ini digunakan untuk traversal maju dari satu node ke node berikutnya.
    - prev: Merupakan referensi ke node sebelumnya dalam linked list. Ini digunakan untuk traversal mundur dari satu node ke node sebelumnya.
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

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

    - head = null: Menginisialisasi linked list dalam keadaan kosong dengan tidak ada node awal (head).
    - size = 0: Mengatur ukuran linked list menjadi nol karena saat pembuatan linked list baru, belum ada node yang ditambahkan.
4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

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

    Saat menambahkan node baru di awal linked list, node tersebut akan menjadi node pertama (head). Karena tidak ada node sebelum node ini, referensi prev diatur menjadi null.
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
    Setelah menambahkan node baru di depan linked list, node baru ini menjadi head yang baru. Statement ini mengatur referensi prev dari node yang sebelumnya menjadi head agar menunjuk ke node baru ini. Hal ini menjaga konsistensi referensi dua arah dalam 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);
```

Saat menambahkan node di akhir linked list, current mengacu pada node terakhir yang ada saat ini. Dengan demikian, prev dari node baru akan menunjuk ke node terakhir ini, dan next diatur ke null karena node baru akan menjadi node terakhir yang baru.

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.

- Loop while (i < index): Melakukan traversal untuk menemukan node pada posisi index.
- if (current.prev == null):
    - Jika current.prev adalah null, berarti current adalah head dari linked list.
    - Membuat node baru (newNode) yang prev-nya adalah null (karena node baru akan menjadi head), data-nya adalah item, dan next-nya adalah current.
    - Mengatur head linked list ke node baru (newNode).
- else:
    - Membuat node baru (newNode) yang prev-nya adalah current.prev (node sebelum current), data-nya adalah item, dan next-nya adalah current.
    - Mengatur prev.next dari node sebelum current ke node baru (newNode).
    - Mengatur prev dari current ke node baru (newNode).

PERCOBAAN 2

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(message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (size == 1) {
        head = null;
    } else {
        head = head.next;
        head.prev = null;
    }
    size--;
}
```

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

```
public void removeLast() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (head.next == null) {
        head = null;
    } else {
        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(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
        Node current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        if (current.next == null) {
            current.prev.next = null;
        } else if (current.prev == null) {
            head = current.next;
            head.prev = null;
        } 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.

```
package doublelinkedlists14;

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

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

        try {
            dll.removeFirst();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println(x:"=========================");

        try {
            dll.removeLast();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println(x:"=========================");

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

Hasil Output:

```
50      40      10      20
Berhasil diisi
Size: 5
==========================
40      10      20
Berhasil diisi
Size: 4
==========================
40      10
Berhasil diisi
Size: 3
==========================
40
Berhasil diisi
Size: 2
==========================
PS D:\Matkul\SEM 2\ASD\Jobsheet11>
```

PERTANYAAN PERCOBAAN

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

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

   - head = head.next;: Statement ini mengubah referensi head ke node berikutnya dalam linked list. Artinya, node pertama saat ini (node lama) dilewati dan node kedua menjadi node pertama (head) yang baru.
   - head.prev = null;: Setelah mengubah head ke node baru, statement ini mengatur referensi prev dari node baru (yang sekarang menjadi head) ke null, karena tidak ada node sebelum head yang baru. Ini penting untuk menjaga konsistensi linked list dua arah.

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
   - while (current.next.next != null): Loop ini digunakan untuk traversal melalui linked list sampai menemukan node kedua terakhir. Kondisi current.next.next != null memastikan bahwa current berhenti pada node kedua terakhir.
   - Setelah loop selesai, current adalah node kedua terakhir dan current.next adalah node terakhir. Dengan mengatur current.next ke null, kita menghapus referensi ke node terakhir, sehingga node kedua terakhir menjadi node terakhir yang baru.
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;
```

Potongan kode ini mengasumsikan bahwa node kedua (head.next) adalah node yang akan dihapus, lalu mengatur head.next ke node ketiga (tmp.next) dan mengatur referensi prev dari node ketiga ke head.
Masalah:

- Kode ini menghapus node kedua tanpa memeriksa apakah linked list memiliki setidaknya dua node.
- Tidak mengubah ukuran (size) linked list.
- Tidak bisa digunakan untuk menghapus node di posisi lain selain node kedua.
- Tidak mengatur ulang referensi prev untuk node yang dihapus (tmp), sehingga memori yang dialokasikan untuk node tersebut mungkin tidak dibersihkan dengan benar (meskipun dalam Java, ini biasanya ditangani oleh garbage collector).

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

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

- current.prev.next = current.next;: Statement ini mengatur referensi next dari node sebelumnya (current.prev) untuk melompati current dan menunjuk ke node berikutnya setelah current. Ini secara efektif mengeluarkan current dari linked list dari sisi sebelumnya.
- current.next.prev = current.prev;: Statement ini mengatur referensi prev dari node berikutnya (current.next) untuk melompati current dan menunjuk ke node sebelum current. Ini secara efektif mengeluarkan current dari linked list dari sisi berikutnya.
- Kedua statement tersebut bekerja bersama-sama untuk menghapus current dari linked list tanpa meninggalkan referensi yang mengarah ke node yang dihapus, memastikan linked list tetap konsisten setelah penghapusan node di tengah

PERCOBAAN 3

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(message:"Linked List kosong");
    }
    return head.data;
}
```

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

```
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List kosong");
    }
    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(message:"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!

```
package doublelinkedlists14;

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

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

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

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

        try {
            System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        try {
            System.out.println("Data akhir pada Linked Lists adalah: " + dll.getLast());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        try {
            System.out.println("Data indeks ke-1 pada Linked Lists adalah: " + dll.get(index:1));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        System.out.println(x:"------------------------------------------");
    }
}
```

Hasil Output:

```
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
------------------------------------------
PS D:\Matkul\SEM 2\ASD\Jobsheet11>
```

Pertanyaan Percobaan

1. Jelaskan method size() pada class DoubleLinkedLists!
    - size adalah atribut dalam class DoubleLinkedLists yang melacak jumlah node dalam linked list.
    - Method size() hanya mengembalikan nilai dari atribut size.
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke- 1!
   Biasanya, indeks pada linked list (seperti pada array) dimulai dari 0. Untuk mengatur indeks agar dimulai dari 1, Anda dapat menyesuaikan logika pada operasi terkait indeks. Namun, ini tidak dianjurkan karena akan membuat kode tidak konvensional dan lebih sulit dimengerti. Berikut adalah contoh bagaimana Anda bisa menyesuaikan method `get(int index)` untuk bekerja dengan indeks yang dimulai dari 1:

```
public int get(int index) throws Exception {
    if (isEmpty() || index < 1 || index > size) {
        throw new Exception(message:"Nilai indeks di luar batas.");
    }
    Node tmp = head;
    for (int i = 1; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

    - Indeks diatur untuk mulai dari 1, sehingga pengecekan batasan indeks (`index < 1` dan `index > size`) disesuaikan.
    - Loop dimulai dari 1 dan berjalan hingga `i < index`, bukan `i < index - 1` seperti dalam implementasi dengan indeks yang dimulai dari 0.
3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!
   DoubleLinkedLists:
    - Setiap node memiliki dua referensi: `prev` (mengarah ke node sebelumnya) dan `next` (mengarah ke node berikutnya).
    - Operasi penambahan node (baik di awal, akhir, atau posisi tertentu) melibatkan pengaturan kedua referensi (`prev` dan `next`), sehingga membutuhkan sedikit lebih banyak pekerjaan dibandingkan single linked list.
    - Penghapusan node juga lebih efisien karena dapat langsung mengakses node sebelumnya.

   SingleLinkedLists:
    - Setiap node hanya memiliki satu referensi: `next` (mengarah ke node berikutnya).
    - Operasi penambahan node (terutama di akhir) mungkin memerlukan traversal dari awal hingga akhir linked list, yang tidak diperlukan dalam double linked list jika ada referensi ke node terakhir.
    - Penghapusan node di posisi tertentu juga membutuhkan traversal karena tidak ada referensi langsung ke node sebelumnya.
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

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

Metode pertama (a):

- Mengecek apakah atribut `size` adalah 0.
- Jika `size` adalah 0, maka linked list dianggap kosong.
- Menggunakan statement `if-else` untuk menentukan nilai boolean yang akan dikembalikan.

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

b.

Metode kedua (b):

- Mengecek apakah referensi `head` adalah `null`.
- Jika `head` adalah `null`, maka linked list dianggap kosong.
- Metode ini lebih ringkas dan langsung dibandingkan metode pertama.

Perbedaan utama:

- Metode pertama memeriksa ukuran linked list melalui atribut `size`, sedangkan metode kedua memeriksa langsung referensi ke node pertama (`head`).
- Metode kedua cenderung lebih sederhana dan langsung karena tidak memerlukan pengecekan kondisi `if-else`.

TUGAS PRAKTIKUM

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)
   DoubleLinkedList.java

vaksin14 > DoubleLinkedLists.java > DoubleLinkedLists > removeFirstAndGetData()

```
package vaksin14;

public class DoubleLinkedLists {
    Node head;
    int size;

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

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

    public void addLast(int nomorAntrian, String nama) {
        if (isEmpty()) {
            head = new Node(prev:null, nomorAntrian, nama, next:null);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, nomorAntrian, nama, next:null);
            current.next = newNode;
        }
        size++;
    }

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

```java
    public void print() {
        if (!isEmpty()) {
            Node tmp = head;
            while (tmp != null) {
                System.out.println(tmp.nomorAntrian + "\t\t" + tmp.nama);
                tmp = tmp.next;
            }
        } else {
            System.out.println(x:"Linked List Kosong");
        }
        System.out.println(x:"------------------------------");
        System.out.println("Sisa Antrian: " + size);
    }

    public String removeFirstAndGetData() throws Exception {
        if (isEmpty()) {
            throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus!");
        } else {
            String nama = head.nama;
            removeFirst();
            return nama;
        }
    }

    public String removeFirstAndGetData() throws Exception {
        if (isEmpty()) {
            throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus!");
        } else {
            String nama = head.nama; // Simpan nama penerima yang akan dihapus
            if (size == 1) {
                head = null;
            } else {
                head = head.next;
                head.prev = null;
            }
            size--;
            return nama; // Kembalikan nama penerima yang dihapus
        }
    }


    public int getSize() {
        return size;
    }
}

```

DoubleLinkedListMain.java

```java
vaksin14 > J DoubleLinkedListsMain.java > DoubleLinkedListsMain > main(String[])
package vaksin14;

import java.util.Scanner;

public class DoubleLinkedListsMain {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedLists dll = new DoubleLinkedLists();

        while (true) {
            System.out.println(x:"++++++++++++++++++++++++++++++");
            System.out.println(x:"PENGANTRI VAKSIN EXTRAVAGANZA");
            System.out.println(x:"1. Tambah Data Penerima Vaksin");
            System.out.println(x:"2. Hapus Data Pengantri Vaksin");
            System.out.println(x:"3. Daftar Penerima Vaksin");
            System.out.println(x:"4. Keluar");
            System.out.println(x:"++++++++++++++++++++++++++++++");
            System.out.print(s:"Pilih menu: ");
            int pilihan = sc.nextInt();
            sc.nextLine();
```

```java
        switch (pilihan) {
            case 1:
                System.out.println("------------------------------");
                System.out.println("Masukkan Data Penerima Vaksin");
                System.out.println("------------------------------");
                System.out.print("Nomor Antrian: ");
                int nomorAntrian = sc.nextInt();
                sc.nextLine(); // Consume newline
                System.out.print("Nama Penerima: ");
                String nama = sc.nextLine();
                dll.addLast(nomorAntrian, nama);
                break;
            case 2:
                try {
                    String namaDivaksinasi = dll.removeFirstAndGetData();
                    System.out.println(namaDivaksinasi + " telah divaksinasi");
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
                break;
            case 3:
                System.out.println("++++++++++++++++++++++++++++++");
                System.out.println("Daftar Pengantri Vaksin");
                System.out.println("++++++++++++++++++++++++++++++");
                System.out.println("No.\t\tNama");
                dll.print();
                System.out.println("(tekan enter untuk melanjutkan)");
                sc.nextLine();
                break;
            case 4:
                System.out.println("Keluar dari program...");
                sc.close();
                return;
            default:
                System.out.println("Pilihan tidak valid!");
        }
```

Node.java

```java
vaksin14 > J Node.java > ...
package vaksin14;

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

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

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.

   Film.java

Film.java > ...

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

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

class DoubleLinkedLists {
    private Film head;
    private int size;

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

    // Method untuk menambah data di awal
    public void addFirst(int id, String judul, double rating) {
        Film newNode = new Film(id, judul, rating);
        if (isEmpty()) {
            head = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    // Method untuk menambah data di akhir
    public void addLast(int id, String judul, double rating) {
        Film newNode = new Film(id, judul, rating);
        if (isEmpty()) {
            head = newNode;
        } else {
            Film current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
            newNode.prev = current;
        }
        size++;
    }
```

```java
    // Method untuk menambah data di indeks tertentu
    public void addAtIndex(int id, String judul, double rating, int index) {
        if (index < 0 || index > size) {
            System.out.println(x:"Invalid index");
            return;
        }
        if (index == 0) {
            addFirst(id, judul, rating);
        } else if (index == size) {
            addLast(id, judul, rating);
        } else {
            Film newNode = new Film(id, judul, rating);
            Film current = head;
            for (int i = 0; i < index - 1; i++) {
                current = current.next;
            }
            newNode.next = current.next;
            current.next.prev = newNode;
            current.next = newNode;
            newNode.prev = current;
            size++;
        }
    }

    // Method untuk mencetak data film
    public void print() {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        System.out.println(x:"==============================");
        System.out.println(x:"DATA FILM LAYAR LEBAR");
        System.out.println(x:"==============================");
        Film current = head;
        int count = 1;
        while (current != null) {
            System.out.println("ID: " + current.id);
            System.out.println("Judul Film: " + current.judul);
            System.out.println("Rating: " + current.rating);
            current = current.next;
            count++;
        }
    }
```

```java
    // Method untuk mencari data berdasarkan ID
    public void searchById(int id) {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        Film current = head;
        int index = 1;
        while (current != null) {
            if (current.id == id) {
                System.out.println("Data Id Film: " + id + " berada di node ke-" + index);
                System.out.println(x:"IDENTITAS:");
                System.out.println("ID Film: " + current.id);
                System.out.println("Judul Film: " + current.judul);
                System.out.println("IMDB Rating: " + current.rating);
                return;
            }
            current = current.next;
            index++;
        }
        System.out.println("Film dengan ID " + id + " tidak ditemukan");
    }

    // Method untuk mengurutkan data berdasarkan rating secara descending
    public void sortRatingDesc() {
        if (isEmpty() || size == 1) {
            return;
        }
        for (Film current = head; current != null; current = current.next) {
            for (Film index = current.next; index != null; index = index.next) {
                if (current.rating < index.rating) {
                    swap(current, index);
                }
            }
        }
    }
```

```java
    private void swap(Film a, Film b) {
        double tempRating = a.rating;
        int tempId = a.id;
        String tempJudul = a.judul;

        a.rating = b.rating;
        a.id = b.id;
        a.judul = b.judul;

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

    // Method untuk mengecek apakah list kosong
    public boolean isEmpty() {
        return size == 0;
    }

    // Method untuk menghapus data pertama
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        if (head.next == null) {
            head = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        size--;
    }
```

Main.java

```java
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoubleLinkedLists doubleLinkedLists = new DoubleLinkedLists();

        int menu;
        do {
            System.out.println(x:"==============================");
            System.out.println(x:"DATA FILM LAYAR LEBAR");
            System.out.println(x:"==============================");
            System.out.println(x:"1. Tambah Data Awal");
            System.out.println(x:"2. Tambah Data Akhir");
            System.out.println(x:"3. Tambah Data Index Tertentu");
            System.out.println(x:"4. Hapus Data Pertama");
            System.out.println(x:"5. Hapus Data Terakhir");
            System.out.println(x:"6. Hapus Data Tertentu");
            System.out.println(x:"7. Cetak");
            System.out.println(x:"8. Cari ID Film");
            System.out.println(x:"9. Urut Data Rating Film-DESC");
            System.out.println(x:"10. Keluar");
            System.out.println(x:"==============================");
            System.out.print(s:"Pilih menu: ");
            menu = scanner.nextInt();
            scanner.nextLine();
```

```
            switch (menu) {
                case 1:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Masukkan Data Film Posisi Awal");
                    System.out.print(s:"ID Film: ");
                    int idFirst = scanner.nextInt();
                    scanner.nextLine(); // consume newline
                    System.out.print(s:"Judul Film: ");
                    String judulFirst = scanner.nextLine();
                    System.out.print(s:"Rating Film: ");
                    double ratingFirst = scanner.nextDouble();
                    doubleLinkedLists.addFirst(idFirst, judulFirst, ratingFirst);
                    break;

                case 2:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Masukkan Data Posisi Akhir");
                    System.out.print(s:"ID Film: ");
                    int idLast = scanner.nextInt();
                    scanner.nextLine(); // consume newline
                    System.out.print(s:"Judul Film: ");
                    String judulLast = scanner.nextLine();
                    System.out.print(s:"Rating Film: ");
                    double ratingLast = scanner.nextDouble();
                    doubleLinkedLists.addLast(idLast, judulLast, ratingLast);
                    break;

                case 3:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Masukkan Data Film di Index Tertentu");
                    System.out.print(s:"ID Film: ");
                    int idIndex = scanner.nextInt();
                    scanner.nextLine(); // consume newline
                    System.out.print(s:"Judul Film: ");
                    String judulIndex = scanner.nextLine();
                    System.out.print(s:"Rating Film: ");
                    double ratingIndex = scanner.nextDouble();
                    System.out.print(s:"Index: ");
                    int index = scanner.nextInt();
                    doubleLinkedLists.addAtIndex(idIndex, judulIndex, ratingIndex, index);
                    break;

                case 4:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Hapus Data Film Pertama");
                    doubleLinkedLists.removeFirst(); // You need to implement this method
                    break;
```

```
                case 4:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Hapus Data Film Pertama");
                    doubleLinkedLists.removeFirst(); // You need to implement this method
                    break;

                case 5:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Hapus Data Film Terakhir");
                    doubleLinkedLists.removeLast(); // You need to implement this method
                    break;

                case 6:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Hapus Data Film di Index Tertentu");
                    System.out.print(s:"Index: ");
                    int removeIndex = scanner.nextInt();
                    doubleLinkedLists.removeAtIndex(removeIndex); // You need to implement this method
                    break;

                case 7:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Cetak Data Film");
                    doubleLinkedLists.print();
                    break;

                case 8:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Cari Film Berdasarkan ID");
                    System.out.print(s:"ID Film: ");
                    int searchId = scanner.nextInt();
                    doubleLinkedLists.searchById(searchId);
                    break;

                case 9:
                    System.out.println(x:"------------------------------");
                    System.out.println(x:"Urutkan Data Film Berdasarkan Rating - DESC");
                    doubleLinkedLists.sortRatingDesc();
                    System.out.println(x:"Data berhasil diurutkan.");
                    break;

                case 10:
                    System.out.println(x:"Keluar...");
                    break;

                default:
                    System.out.println(x:"Pilihan tidak valid. Silakan coba lagi.");
            }
        } while (menu != 10);

        scanner.close();
    }
}
```

```java
    // Method untuk menghapus data terakhir
    public void removeLast() {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        if (head.next == null) {
            head = null;
        } else {
            Film current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.prev.next = null;
        }
        size--;
    }

    // Method untuk menghapus data di index tertentu
    public void removeAtIndex(int index) {
        if (index < 0 || index >= size) {
            System.out.println(x:"Invalid index");
            return;
        }
        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--;
        }
    }
}
```

```java
    private void swap(Film a, Film b) {
        double tempRating = a.rating;
        int tempId = a.id;
        String tempJudul = a.judul;

        a.rating = b.rating;
        a.id = b.id;
        a.judul = b.judul;

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

    // Method untuk mengecek apakah list kosong
    public boolean isEmpty() {
        return size == 0;
    }

    // Method untuk menghapus data pertama
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        if (head.next == null) {
            head = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        size--;
    }
```

```java
    // Method untuk menghapus data terakhir
    public void removeLast() {
        if (isEmpty()) {
            System.out.println(x:"Data film kosong");
            return;
        }
        if (head.next == null) {
            head = null;
        } else {
            Film current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.prev.next = null;
        }
        size--;
    }

    // Method untuk menghapus data di index tertentu
    public void removeAtIndex(int index) {
        if (index < 0 || index >= size) {
            System.out.println(x:"Invalid index");
            return;
        }
        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--;
        }
    }
}
```

Hasil Output:

```
================================
DATA FILM LAYAR LEBAR
================================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
================================
Pilih menu: 1
--------------------------------
Masukkan Data Film Posisi Awal
ID Film: 1444
Judul Film: Spiderman: No way home
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
------------------------------
Masukkan Data Posisi Akhir
ID Film: 16969
Judul Film: Boku no pico
Rating Film: 3.3
```

```
==============================
DATA FILM LAYAR LEBAR
==============================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==============================
Pilih menu: 3
------------------------------
Masukkan Data Film di Index Tertentu
ID Film: 1234
Judul Film: Death on the Nile
Rating Film: 6.6
Index: 3
Invalid index
```

```
==============================
DATA FILM LAYAR LEBAR
==============================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==============================
Pilih menu: 7
------------------------------
Cetak Data Film
==============================
DATA FILM LAYAR LEBAR
==============================
ID: 1444
Judul Film: Spiderman: No way home
Rating: 8.7
ID: 16969
Judul Film: Boku no pico
Rating: 3.3
```