JOBSHEET 12
Double Linked Lists

12.1 Kegiatan Praktikum 1

12.2.1 Percobaan 1

- **DoubleLinkedList**

```
package doublelinkedlist;

public class DoubleLinkedList10 {
    Node10 head;
    int size;

public DoubleLinkedList10(){
    head = null;
    size = 0;
}
public boolean isEmpty(){
    return head == null;
}
public void addFirst(int item) {
if (isEmpty()){
    head = new Node10(prev:null, item, next:null);
} else {
    Node10 newNode = new Node10(prev:null, item, head);
    head.prev = newNode;
    head = newNode;
}
size++;
}
public void addLast(int item){
    if(isEmpty()){
        addFirst(item);
    } else {
        Node10 current = head;
        while (current.next !=null){
            current = current.next;
        }
        Node10 newNode = new Node10 (current, item, next: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(message:"Nilai indeks di luar batas");
  } else {
      Node10 current = head;
      int i = 0;
      while (i < index) {
          i++;
      }
      if(current.prev == null) {
          Node10 newNode = new Node10 (prev:null, item, current);
          current.prev = newNode;
          head = newNode;
      } else {
          Node10 newNode = new Node10 (current.prev, item, current);
          newNode.prev = current.prev;
          newNode.next = 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()){
      Node10 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");
  }
}
```

- **DoubleLinkedListMain**

```
package doublelinkedlist;

public class DoubleLinkedListMain10 {
    Run | Debug
    public static void main (String[] args) {
        DoubleLinkedList10 dll = new DoubleLinkedList10();
        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) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        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

```
Linked Lists Kosong
Size :0
====================
7       3       4
berhasil diisi
Size: 3
====================
40      7       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!
    - Dari struktur node, setiap single linked lst memiliki dua komponen utama yaitu data dan pointer/reference ke node beriktnya, sedangkan doube linked list memiiki 3 kompone utama yaitu data, pointer ke node berikutnya dan pointer ke node sebelumnya
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
    - Atribut next
        - digunakan untuk memnunjuk ke node berikutnya dalam list
        - Setiap node memiliki pointer next yang menunjukkan ke node berikutnya
        - Pointer next pada node terakhir (tail) memiliki nilai null karema tidak ada node lain di belakangnya
    - Atribut prev
        - Digunakan untuk menunjukkan ke node sebelumnya dalam list
        - Setiap node memiliki poiner prev yang menunjuk ke node sebelumnya
        - Pointer prev pada node pertama (head) memiliki nilai null karena tidak ada node lain didepannya.
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dansize seperti pada gambar berikut ini?

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

    - Menginisialisasi list kosong : memberikan nilai awal yang sesuai untuk atribut head dan size yang merepresentasikan keadaan awal list yang kosong
    - Menyimpan informasi awal list : Menyimpan informasi penting tentang struktur dan jmlah node dalam list, yang nantinya akan diperbarui saat operasi seperti penambahan, penyisipan, dan penghapusan node dilakukan

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

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

    - Penetapan prev menjadi null pada method addFirst() merupakan bagian penting dalam implementasi double linked list

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

    - Pada method addFirst() dalam class DoubleLinkedList, statement head.prev = newNode; memiliki peran penting dalam menghubungkan node baru yan ditambahkan ke awal list dengan node yang sebelumnya menjadi head

6. Perhatikan isi method **addLast()**, apa arti dari pembuatan object Node dengan mengisikanparameter prev dengan current, dan next dengan null?

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

    - Pembuata object Node dengan prev = current dan next = null pada method addLast() memastikan struktur dan koneksi dua arah dalam double linked list tetap terjaga saat menambhakan node baru ke akhir list

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

jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

```
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;
}
```

    - Kode dalam kotak kuning tersebut bertanggung jawab untuk membuat node baru mengatur pointer prev dan next nya dan memasukkan ke dalam double linked list di bagian awal

12.2 Kegiatan Praktikum 2

12.3.1 Tahapan Percobaan

- **DoubleLinkedList.java**

```
public void removeFirst() throws Exception {
    if(isEmpty()){
        throw new Exception (message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (size == 1) {
        removeFirst();
    } else {
        head = head.next;
        head.prev = null;
        size--;
    }
}
public void removeLast() throws Exception {
    if(isEmpty()){
        throw new Exception (message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (head.next == null) {
        head = null;
        size--;
        return;
    }
    Node10 current = head;
    while (current.next.next != null){
        current = current.next;
    }
    current.next = null;
    size--;
}
    public void remove (int index) throws Exception {
        if(isEmpty() || index >= size) {
            throw new Exception(message:"Nilai indeks di luar batas");
        } else if (index == 0){
            removeFirst();
        }else {
            Node10 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--;
        }
    }
}
```

- **DoubleLinkedListMain.java**

```
        dll.addLast(item:50);
        dll.addLast(item:40);
        dll.addLast(item:10);
        dll.addLast(item:20);
        dll.print();
        System.out.println("Size: "+dll.size());
        System.out.println(x:"====================");
        dll.removeFirst();
        dll.print();
        System.out.println("Size: "+dll.size());
        System.out.println(x:"======================");
        dll.removeLast();
        dll.print();
        System.out.println("Size: "+dll.size());
        System.out.println(x:"======================");
        dll.remove(index:1);
        dll.print();
        System.out.println("Size: "+dll.size());
    }
```

12.3.2 Verifikasi Hasil Percobaan

```
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;
```

   - merupakan bagian penting dalam implementasi double linked list. Statement ini memastikan penghapusan node pertama (head) dengan benar dan menjaga struktur list yang valid
2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method **removeLast()**?
   - Memeriksa keberadaan node:
   - Menavigasi ke node terakhir
   - Memeriksa data
   - Menghapus node terakhir
   -

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 program yang ditunjukkan tidak cocok untuk peritah remove karena tidak dapat memenuhi semua persyaratan fungsioalitas yang diperlukan

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 pada linked list stelah node dihapus

12.3 Kegiatan Praktikum 3

12.4.1 Tahapan Percobaan

- **DoubleLinkedList.java**

```
public int getFirts() throws Exception {
    if(isEmpty()) {
        throw new Exception (message:"Linked List kosong");
    }
    return head.data;
}
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception (message:"Linked List Kosong");
    }
    Node10 tmp = head;
    while (tmp.next != null){
        tmp = tmp.next;
    }
    return tmp.data;
}
public int get (int index) throws Exception{
    if(isEmpty() || index >=size) {
        throw new Exception(message:"Nilai indeks di luar batas.");
    }
    Node10 tmp = head;
    for(int i = 0; i < index; i++){
        tmp = tmp.next;
    }
    return tmp.data;
}
```

- **DoubleLinkedistMain.java**

```
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("Siize: "+dll.size());
System.out.println(x:"=========================");
System.out.println("Data awal pada Linked List adalah: " + dll.getFirts());
System.out.println("Data akhir pada Linked List adalah: " + dll.getLast());
System.out.println("Data indeks ke-1 pada Linked List adalah: " + dll.get(index:1));
```

12.4.2 Verfikasi Hasil Percobaan

```
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
40
berhasil diisi
Size: 1
=========================
7       3       40      4
berhasil diisi
Size: 4
=============================
40      7       3       40      4
berhasil diisi
Siize: 5
-------------------------
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method **size()** pada class DoubleLinkedLists!
    - Methode size() pada class DoubleLinkedList digunakan untuk mengembalikan jumlah elemen yang terdapat di dalam double linked list
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!
    - Dengan cara menambahkan node dummy di awal, salah satu cara paling sederhana adalah dengan menambah node dummy di awal list. Node dummy ini tidak memiliki data, tetapi berfungsi sebagai penanda untuk memulai indeks dari 1
3. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
   Posisi Penambahan
    - DLL : elemen baru yang dapat ditambahkan di tiga posisi:
        - Depan(head) : Menyisipkan elemen di awal list
        - Belakang (tail) : Menyisipkan elemen di akhir list
        - Tengah : Menyisipkan elemen diantara dua node yang ada
    - SLL : elemen baru hanya dapat ditambahkan di dua posisi:
        - Depan(head) : Menyisipkan elemen diawal list
        - Belakang (tail) : Menyisipkan elemen diakhir list

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

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

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

- Program a
  Menggunakan operator == untuk membandingkan nilai size dengan 0. Jika size sama dengan 0 , makalist kosong. Lalu mengembealikan nilai true jika size nya kosong
- Program b
  Langsung mengcek nilai head. Jika head nya bernilai null, maka list kosong kemudian langsung mengembalikan nilai true

12.5 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 yangtelah divaksinasi di menu Hapus Data(2) harus ada)**

Ilustrasi Program

Menu Awal dan Penambahan Data

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

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

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

 1. Tambah Data Penerima Vaksin
 2. Hapus Data Pengantri Vaksin
 3. Daftar Penerima Vaksin
 4. Keluar
++++++++++++++++++++++++++++++
1
------------------------------
Masukkan Data Penerima Vaksin
------------------------------
Nomor Antrian:
123
-Nama Penerima:
Joko
```

*Cetak Data (**Komponen di area merah harus ada**)*

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

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

*Hapus Data (**Komponen di area merah harus ada**)*

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

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

JOBSHEET12 > Tugas1 > J Node10.java > ...

```java
package Tugas1;


public class Node10 {
    Vaksinasi10  data;
    Node10 prev, next;

    Node10(Node10 prev, Vaksinasi10 data, Node10 next)
    {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

JOBSHEET12 > Tugas1 > J Vaksinasi10.java > Vaksinasi10

```java
package Tugas1;

public class Vaksinasi10 {
    String nama;
    int noAntri;

    Vaksinasi10(){}

    Vaksinasi10 (int noAntri, String nama) {
        this.noAntri = noAntri;
        this.nama = nama;
    }

}
```

```java
package Tugas1;

import java.util.Scanner;

public class VaksinasiMain10 {
    public static void menu() {
        System.out.println("++++++++++++++++++++++++++++++++++ ");
        System.out.println("Vaksin 2024");
        System.out.println("++++++++++++++++++++++++++++++++++ ");
        System.out.println();
        System.out.println("1. Tammbah 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.println("++++++++++++++++++++++++++++++++++ ");
    }

    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedList10 dll = new DoubleLinkedList10();

        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();
                    Vaksinasi10 nb = new Vaksinasi10(noAntrian, nama);
                    dll.addlast(nb);
                    System.out.println();
                    break;
                case 2:
                    Vaksinasi10 penerima = dll.getFirst();
                    System.out.println(penerima.nama + " telah selesain divaksinasi.");
                    dll.removeFirst();
                    break;
```

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
```

```
==================================
DATA FILM LAYAR LEBAR
==================================
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film-DESC
 10. Keluar
==================================
2
Masukkan Data Posisi Akhir
ID Film:
1346
Judul Film:
Uncharted
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
==================================
3
Masukkan Data Film
Urutan ke-
ID Film:
1234
Judul Film:
Death on the Nile
Rating Film:
6.6
Data Film ini akan masuk di urutan ke-
3
```

Cetak 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
==================================
7
Cetak Data
ID: 1222
 Judul Film: Spider-Man: No Way Home
 ipk: 8.7
ID: 1765
 Judul Film: Skyfall
 ipk: 7.8
ID: 1567
 Judul Film: The Dark Knight Rises
 ipk: 8.4
ID: 1234
 Judul Film: Death on The Nile
 ipk: 6.6
ID: 1346
 Judul Film: Uncharted
 ipk: 6.7
```

Pencarian Data

```
==================================
DATA FILM LAYAR LEBAR
==================================
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film-DESC
 10. Keluar
==================================
8
Cari Data
Masukkan ID Film yang dicari
1567
Data Id Film: 1567 berada di node ke- 3
IDENTITAS:
 ID Film: 1567
 Judul Film: The Dark Knight Rises
 IMDB Rating: 8.4
```

JOBSHEET12 > Tugas2 > DoubleLinkedLists08.java > DoubleLinkedLists08 > isEmpty()

```java
package Tugas2;

public class DoubleLinkedLists08 {
    Node10 head;
    int size;

    public DoubleLinkedLists08() {
        head = null;
        size = 0;
    }

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

    public void addFirst(Film10 item) {
        if (isEmpty()) {
            head = new Node10(prev:null, item, next:null);
        } else {
            Node10 newNode = new Node10(prev:null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;

    }

    public void addlast(Film10 item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node10 current = head;
            while (current.next != null) {
                current = current.next;
            }

            Node10 newNode = new Node10(current, item, next:null);
            current.next = newNode;
            size++;
        }
    }
```

JOBSHEET12 > Tugas2 > Film10.java > Film10

```java
package Tugas2;

public class Film10 {
    int id;
    String judul;
    double rating;

    Film10(){}

    Film10 (int id, String judul, double rating) {
        this.id = id;
        this.judul = judul;
        this.rating = rating;
    }
}
```

JOBSHEET12 > Tugas2 > FilmMain08.java > FilmMain08 > menu()

```java
package Tugas2;

import java.util.Scanner;

import Tugas1.Vaksinasi10;

public class FilmMain08 {
    public static void menu() {
        System.out.println(x:"---------------------------------- ");
        System.out.println(x:"DATA FILM LAYAR LEBAR 10");
        System.out.println(x:"---------------------------------- ");
        System.out.println();
        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 Terkahir");
        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:"---------------------------------- ");
    }

    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedLists08 dll = new DoubleLinkedLists08();

        int pilih;

        do {
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    System.out.println(x:"----------------------------");
                    System.out.println(x:"Masukkan Data Posisi Awal");
                    System.out.println(x:"----------------------------");
                    System.out.print(s:"ID : ");
                    int id = sc.nextInt();
                    System.out.print(s:"Judul Film : ");
                    String judul = sc.next();
                    System.out.print(s:"Rating (ex. 4.5) : ");
                    double rating = sc.nextDouble();
```

JOBSHEET12 > Tugas2 > Node10.java > Node10

```java
package Tugas2;

// import org.w3c.dom.Node;

public class Node10 {
    Film10  data;
    Node10 prev, next;

    Node10(Node10 prev, Film10 data, Node10 next)
    {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```