Nama : Ahmad Franklyn Bima Aquilla

Kelas : SIB 1F – 01

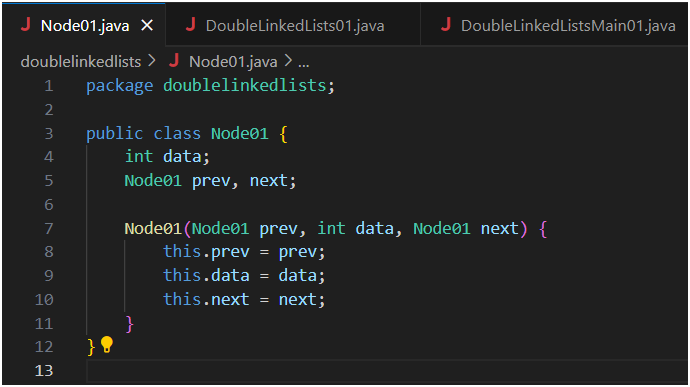
NIM : 2341760027

Matkul : Praktikum Algoritma dan Struktur Data

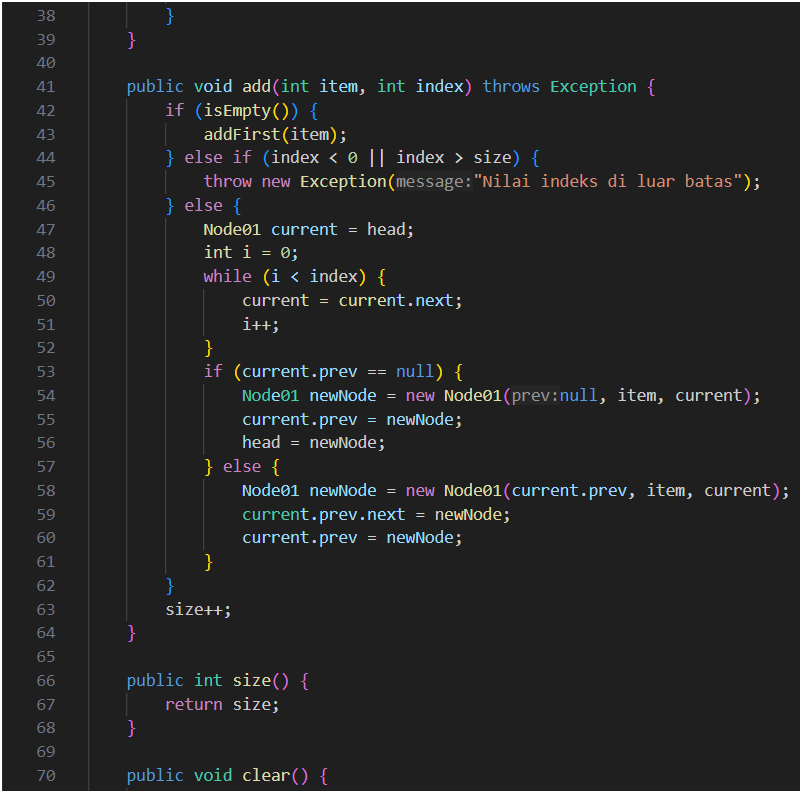
Github : <https://github.com/JustQuill25/Jobsheet12PraktikumASD>

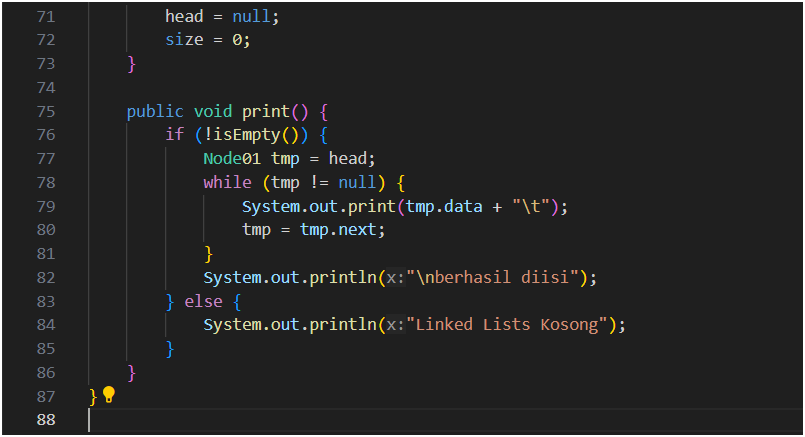
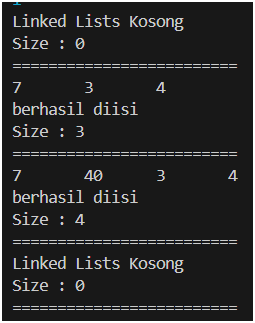
**PERTEMUAN 12**

**(** ASD – Double *Linked List* **)**

***Class Node01 Class DoubleLinkedLists01***

***Class DoubleLinkedListsMain01***



***Output***

## Pertanyaan Percobaan

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

* Perbedaan utama antara single linked list dan double linked list terletak pada jumlah pointer yang dimiliki oleh setiap node dan arah traversal yang didukung oleh list tersebut. Double linked list menawarkan fleksibilitas lebih dalam hal traversal dan penghapusan node, namun dengan biaya memori yang lebih tinggi dan kompleksitas implementasi yang lebih besar.

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

* Atribut **next** dan **prev** dalam class **Node01** memungkinkan linked list melakukan traversal dua arah. **next** memungkinkan pergerakan maju, sedangkan **prev** memungkinkan pergerakan mundur, memberikan fleksibilitas yang lebih besar dibandingkan dengan single linked list.

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

* Pada class **DoubleLinkedLists01**, konstruktor bertugas untuk menginisialisasi objek baru dari linked list. Inisialisasi atribut **head** dan **size** dalam konstruktor tersebut memiliki tujuan untuk memastikan bahwa linked list dimulai dalam keadaan kosong dan teratur

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

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

* Alasan prev Dianggap Sama dengan null
  + - 1. **Node Baru di Depan Linked List:**

Method addFirst() digunakan untuk menambah node baru di bagian depan linkedlist. Node baru ini akan menjadi node pertama atau head dari linked list.

* + - 1. **Node Pertama Tidak Memiliki Node Sebelumnya:**

Dalam double linked list, node pertama (head) tidak memiliki node

sebelumnya. Oleh karena itu, atribut prev dari node pertama harus

null untuk menunjukkan bahwa tidak ada node sebelum node ini.

* + - 1. **Konstruktor Node01:**

Konstruktor Node01 memiliki parameter prev, data, dan next. Ketika menambahkan node baru di depan linked list, prev harus diinisialisasi dengan null karena tidak ada node yang mendahuluinya. next diatur ke node yang sebelumnya adalah head, karena node baru akan menjadi head baru dan node yang sebelumnya head akan menjadi node berikutnya setelah node baru.

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

* Statement **head.prev = newNode** dalam method **addFirst()** menghubungkan node yang saat ini menjadi head dengan node baru yang akan menjadi head. Ini memastikan bahwa node yang sebelumnya head sekarang memiliki referensi ke node baru sebagai node sebelumnya, yang penting untuk mempertahankan struktur dan konsistensi double linked list.

1. 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**);

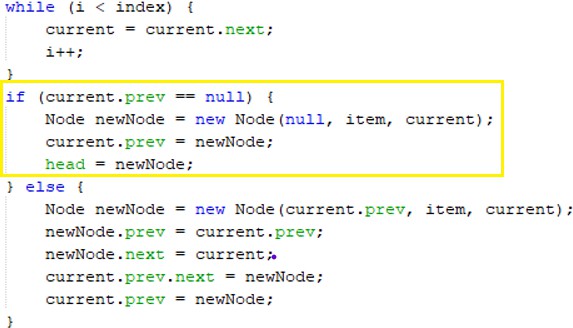
* Statement Node newNode = new Node(current, item, null); dalam method addLast() bertujuan untuk:

1. Menghubungkan node baru dengan node terakhir sebelum penambahan melalui parameter prev.

2. Menetapkan node baru sebagai node terakhir dengan parameter next diatur ke null.

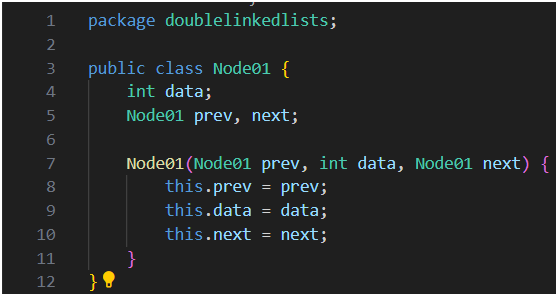
* Ini memastikan bahwa node baru ditambahkan di akhir linked list dengan referensi yang benar ke node sebelumnya dan dengan tidak ada node setelahnya, menjaga struktur double linked list tetap konsisten

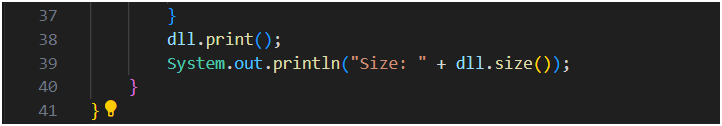
1. Pada method **add(),** terdapat potongan kode program sebagai berikut:

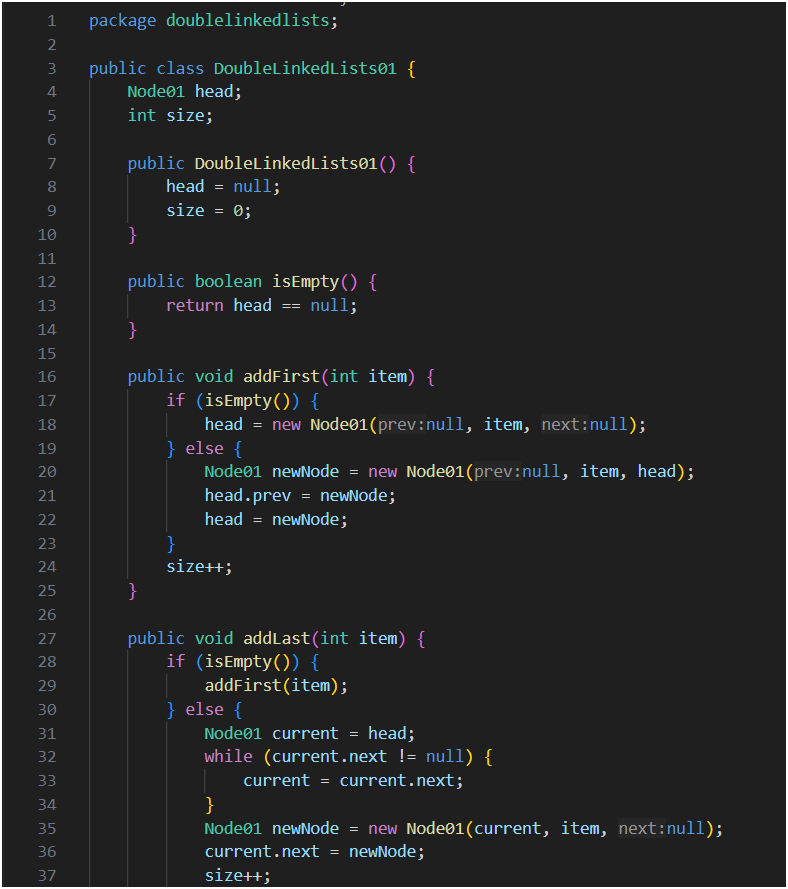


jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

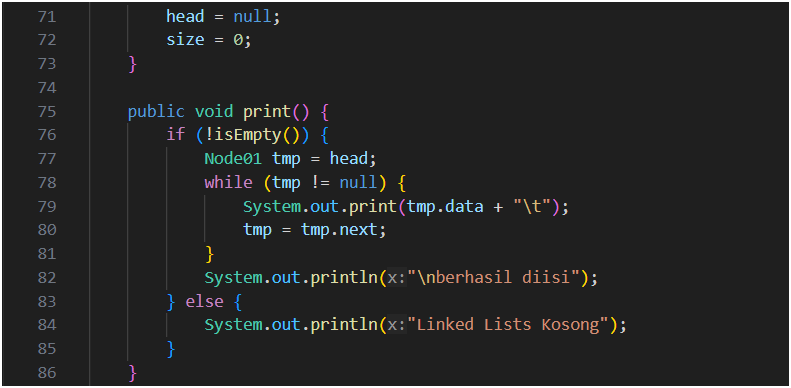
* Bagian kode **if (current.prev == null)** dalam method **add(int item, int index)** digunakan untuk menangani penambahan node baru di depan linked list (posisi 0). Kondisi ini memastikan bahwa jika node saat ini adalah head (karena **prev**-nya **null**), node baru dibuat dan ditempatkan sebelum node saat ini, lalu head diperbarui untuk menunjuk ke node baru. Hal ini menjaga struktur linked list dan memastikan bahwa node baru ditempatkan dengan benar di posisi paling depan.

***Class Node01 Class DoubleLinkedListsMain01***

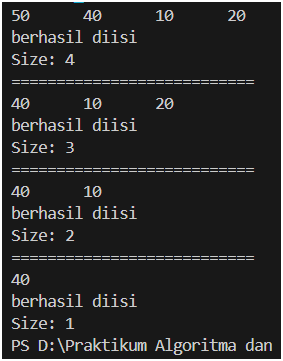
******

***Class DoubleLinkedLists01***







****** ***Output***

## Pertanyaan Percobaan

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

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

* Method **removeFirst()** digunakan untuk menghapus node pertama dari double

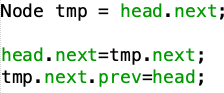
linked list. Jika linked list hanya memiliki satu node, method ini akan memanggil

method **removeLast()** untuk menghapus node tersebut.

* + - 1. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method

**removeLast()**?

* Dalam method **removeLast()**, posisi data terakhir dideteksi dengan cara menelusuri linked list hingga mencapai node kedua terakhir (**current**), yaitu node yang **next.next**-nya adalah **null**. Setelah itu, **next** dari node kedua terakhir diset menjadi **null**, sehingga node terakhir dihapus dari linked list.
  + - 1. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah **remove**!



* Potongan kode **Node01 tmp = head.next; head.next = tmp.next; tmp.next.prev =**

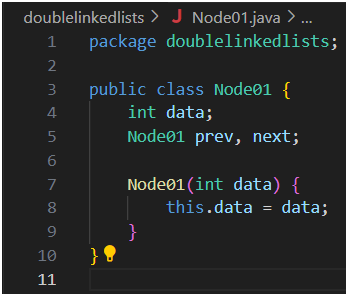
**head;** tidak cocok untuk operasi **remove** karena terbatas pada penghapusan node kedua dalam linked list yang memiliki setidaknya dua node. Kode ini tidak menangani berbagai skenario penghapusan, termasuk node pertama, node terakhir, atau node pada posisi lain, serta tidak memperbarui ukuran linked list dengan benar. Sebuah implementasi **remove** yang tepat harus lebih komprehensif dan mempertimbangkan berbagai kasus untuk memastikan konsistensi dan integritas struktur data.

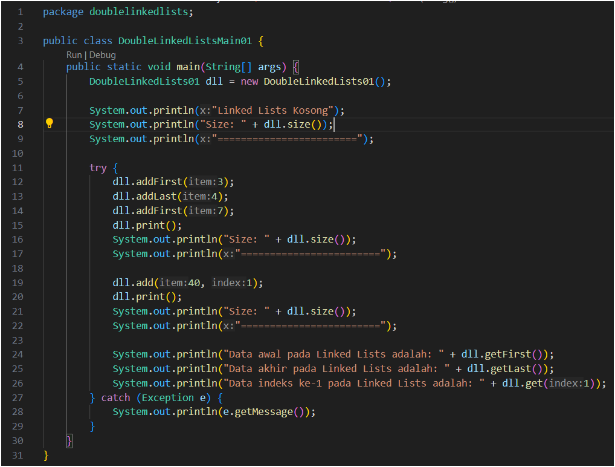
* + - 1. Jelaskan fungsi kode program berikut ini pada fungsi **remove!**

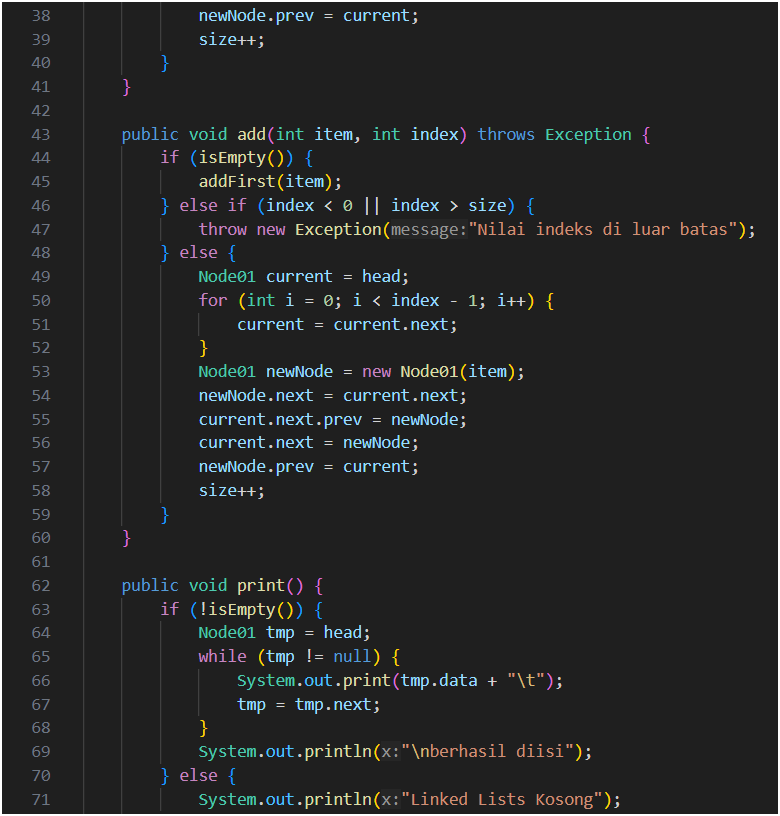


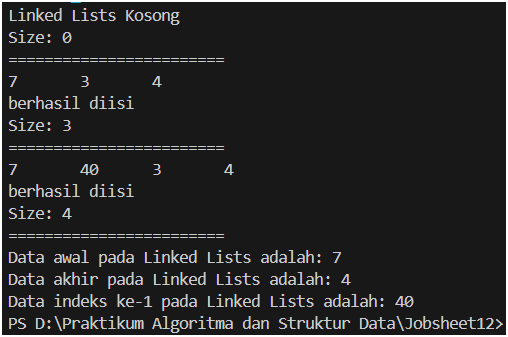
* Potongan kode **current.prev.next = current.next;** dan **current.next.prev =**

**current.prev;** pada fungsi **remove** bertujuan untuk menghapus node yang ditunjuk oleh **current** dari linked list dengan memperbarui tautan **next** dan **prev** di sekitarnya.

***Class Node01 Class DoubleLinkedLists01***

***Class DoubleLinkedListsMain01***

******

***Output***



## Pertanyaan Percobaan

* + - 1. Jelaskan method **size()** pada class DoubleLinkedLists!
* Method **size()** pada class **DoubleLinkedLists** digunakan untuk mengembalikan jumlah

elemen yang ada dalam linked list

* + - 1. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke- 1!
* Dalam double linked list, mengatur indeks untuk dimulai dari 1 (bukan 0) memerlukan penyesuaian pada implementasi method-method yang terlibat dalam pengaksesan atau manipulasi data

1. **Penyesuaian pada Method Add**

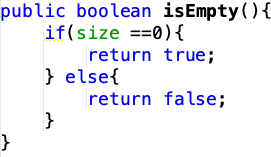
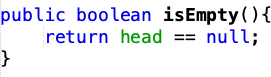
Saat menambahkan elemen pada indeks tertentu, kita perlu memperhitungkan perbedaan antara indeks pengguna (dimulai dari 1) dan indeks internal (dimulai dari 0).

1. **Penyesuaian pada Method Get**

Ketika mengakses elemen pada indeks tertentu, kita perlu mengonversi indeks pengguna menjadi indeks internal. Misalnya, jika pengguna ingin mengambil elemen pada indeks ke-1, maka dalam implementasi method get kita harus mengambil elemen pada indeks 0 dalam linked list.

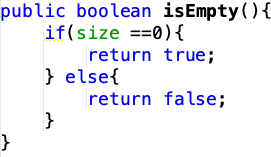
1. **Penyesuaian pada Method Remove**

Saat menghapus elemen pada indeks tertentu, kita juga perlu memperhitungkan perbedaan antara indeks pengguna dan indeks internal. Misalnya, jika pengguna ingin menghapus elemen pada indeks ke-1, maka dalam implementasi method remove kita harus menghapus elemen pada indeks 0 dalam linked list.

* + - 1. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
* Perbedaan karakteristik fungsi **add** pada double linked list dan single linked list terletak pada cara penambahan elemen dilakukan dan kompleksitas waktunya.
* Double linked list memiliki kompleksitas waktu yang lebih baik untuk operasi addFirst dan addLast dibandingkan single linked list. Namun, implementasi double linked list memerlukan pengelolaan tambahan terkait dengan pointer prev, sehingga membutuhkan lebih banyak ruang penyimpanan.
  + - 1. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

( a ) ( b )

* Overall Kedua kode tersebut memiliki fungsi dan tujuan yang sama. Perbedaan antara 2 kode tersebut :



* **Logika** : Kode program ini menggunakan kondisional **if** untuk mengecek apakah **size**

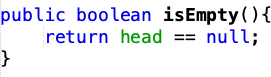
dari linked list sama dengan 0. Jika **size** sama dengan 0, maka dikembalikan

nilai **true** yang menandakan linked list kosong. Jika **size** tidak sama dengan

0, maka dikembalikan nilai **false**.

* **Penjelasan** : Kode program ini secara eksplisit memeriksa nilai **size** untuk menentukan

apakah linked list kosong atau tidak.



* **Logika** : Kode program ini menggunakan operator perbandingan untuk mengecek

apakah variabel **head** bernilai **null**. Jika **head** bernilai **null**, maka

dikembalikan nilai **true** yang menandakan linked list kosong. Jika **head**

tidak bernilai **null**, maka dikembalikan nilai **false**.

* **Penjelasan** : Kode program ini memeriksa apakah **head**, yaitu referensi ke node pertama

dalam linked list, bernilai **null**. Jika **head** bernilai **null**, artinya linked list

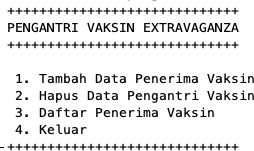
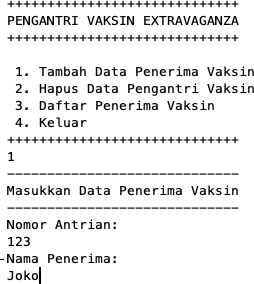
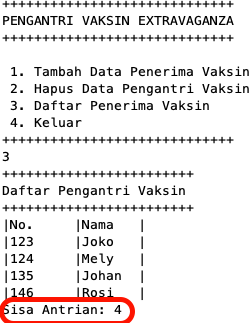
tidak memiliki elemen, sehingga dikembalikan nilai **true**. Jika **head** tidak

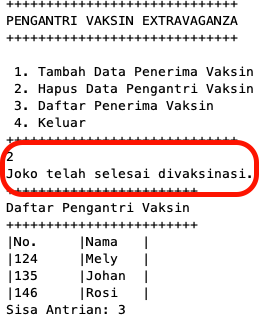
bernilai **null**, artinya linked list memiliki minimal satu elemen, sehingga

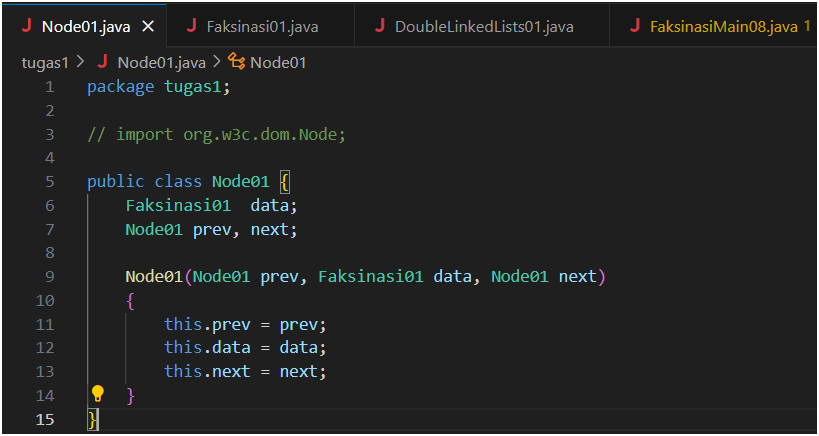
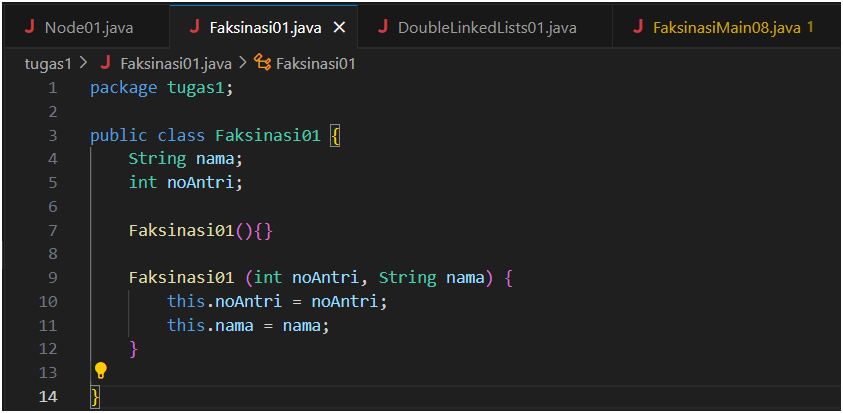
dikembalikan nilai **false**.

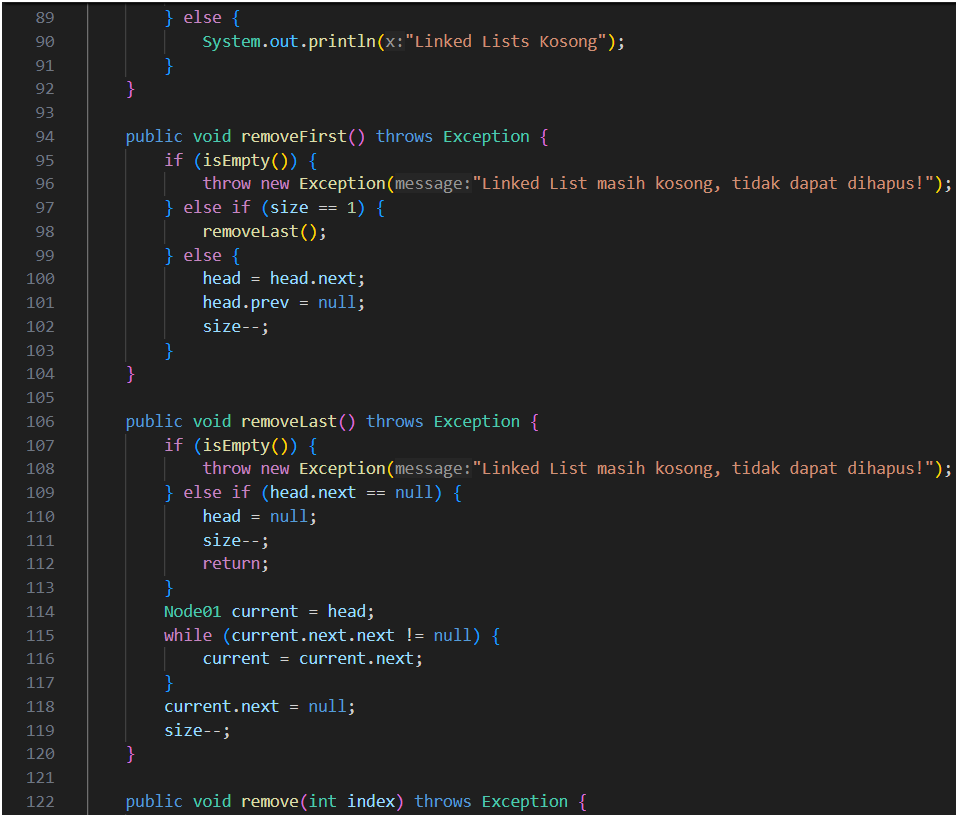
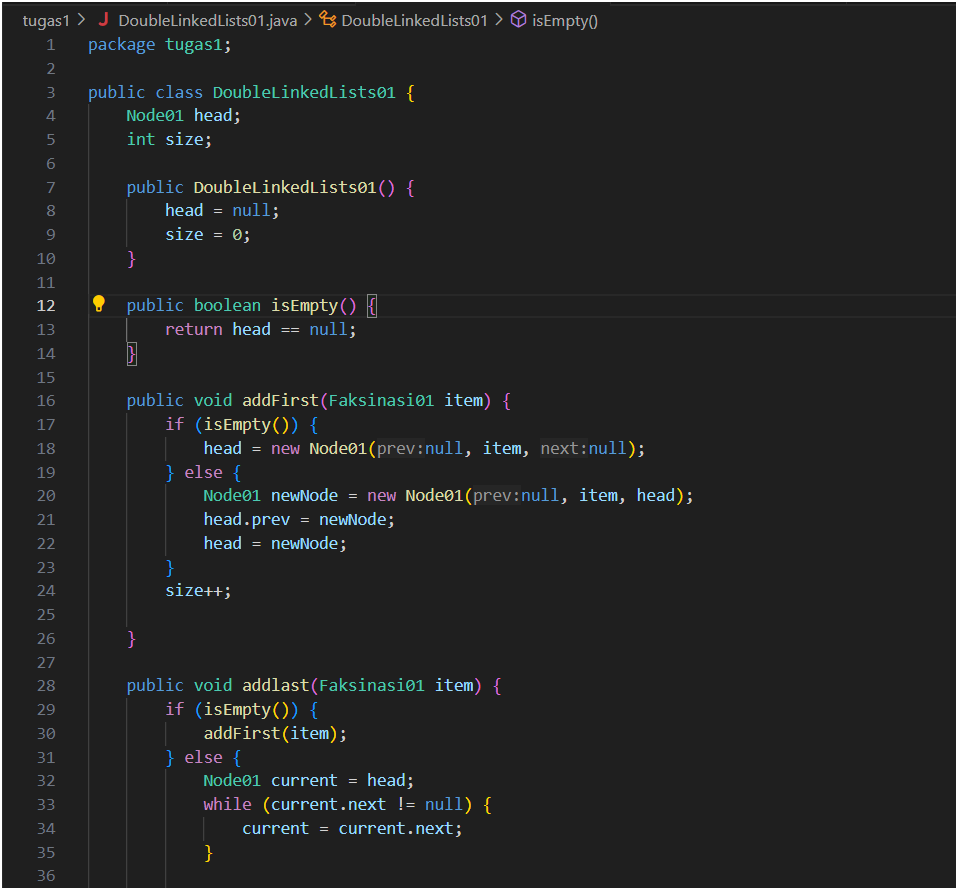
## Tugas Praktikum

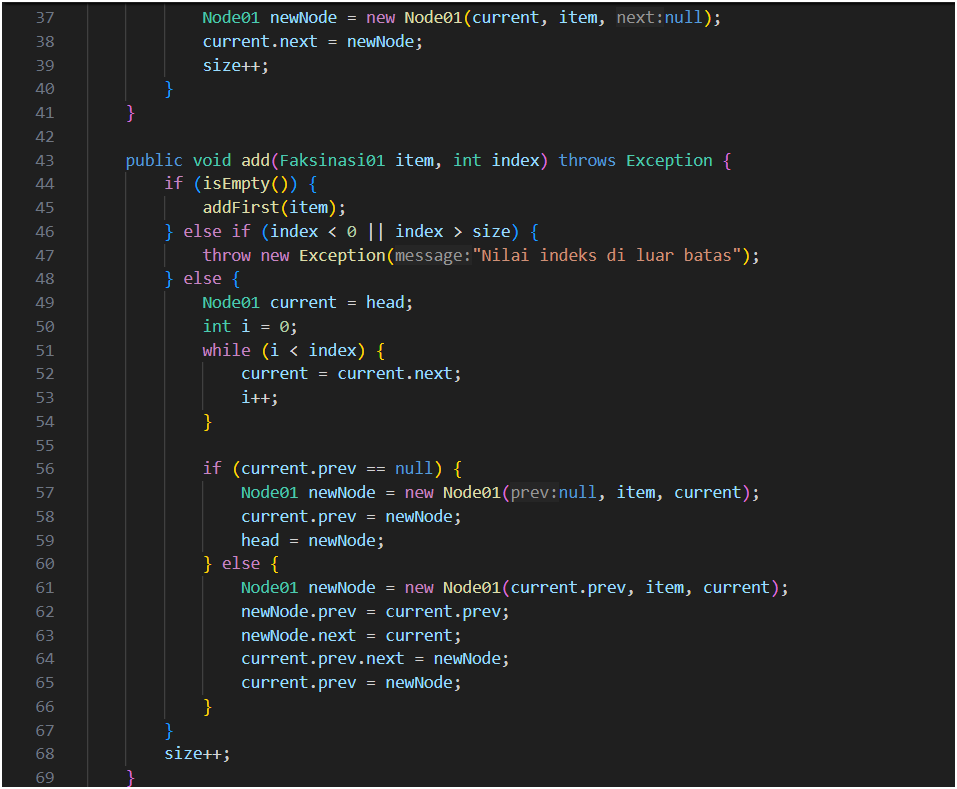
1. Buat program antrian vaksinasi menggunakan queue berbasis double linked list sesuai ilustrasi dan menu di bawah ini! **(counter jumlah antrian tersisa di menu cetak(3) dan data orang yang telah divaksinasi di menu Hapus Data(2) harus ada)**

* *Menu Awal dan Penambahan Data*
* *Cetak Data (****Komponen di area merah harus ada****)*
* *Hapus Data (****Komponen di area merah harus ada****)*

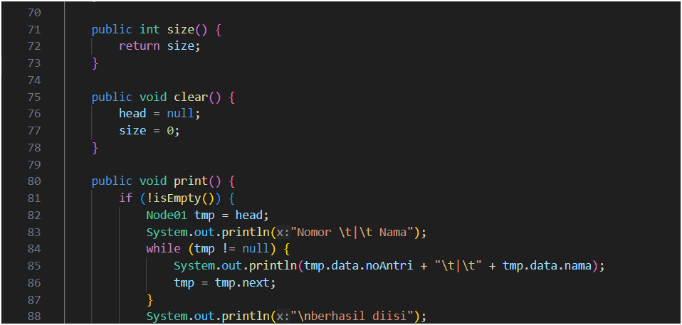
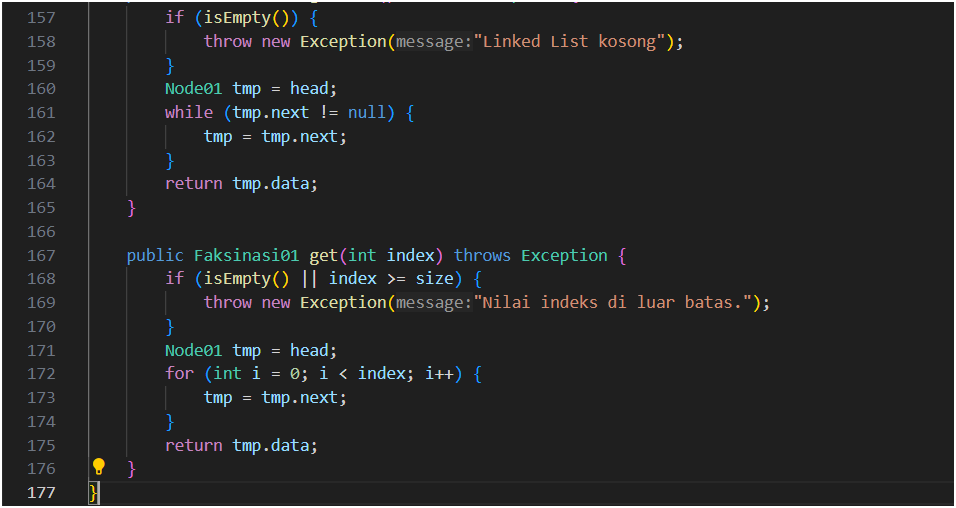


***Class Node01 Class Faksinasi01***

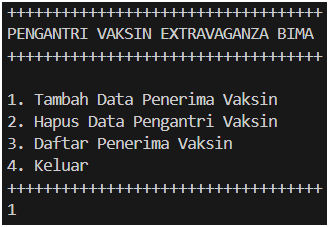
*********Class DoubleLinkedLists01***

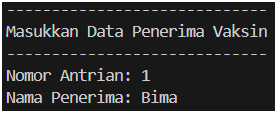


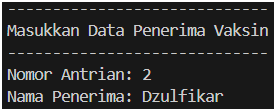
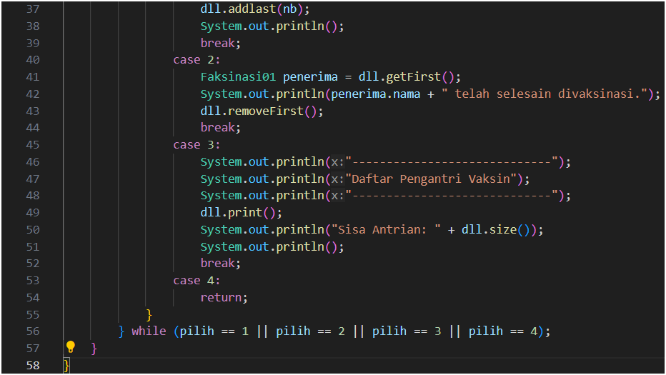


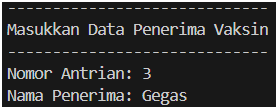


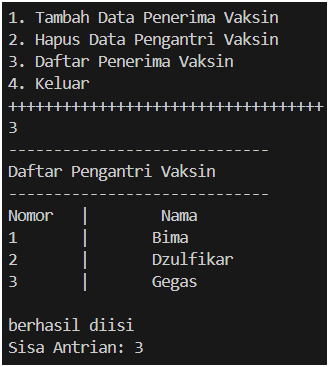
***Class VaksinasiMain01 Output***

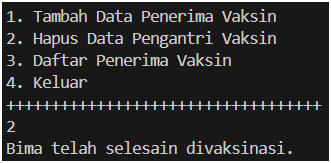
****** *Menambah Data Penerima*



******

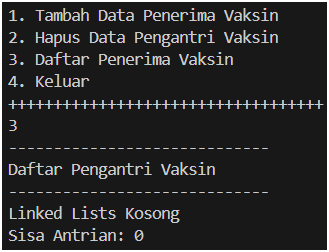


*Daftar Penerima Vaksin*

 *Hapus Data Vaksinasi*

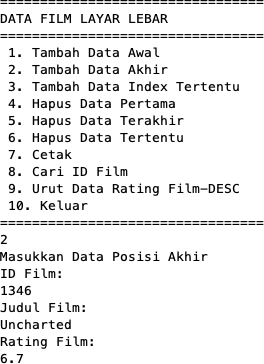




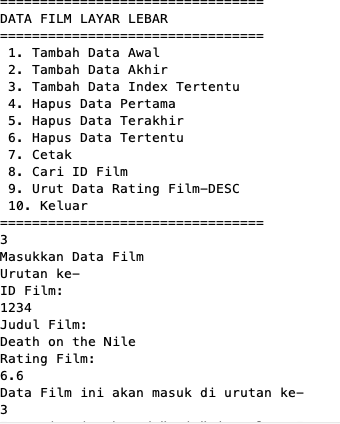


1. Buatlah program daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists, bentuk program memiliki fitur pencarian melalui ID Film dan pengurutan Rating secara descending. Class Film wajib diimplementasikan dalam soal ini.

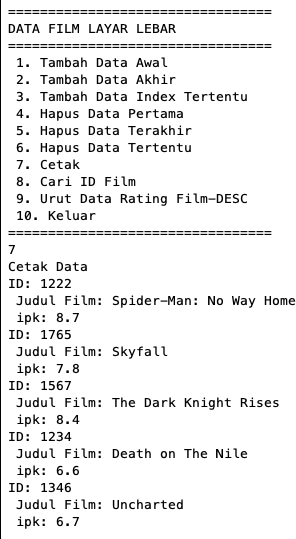
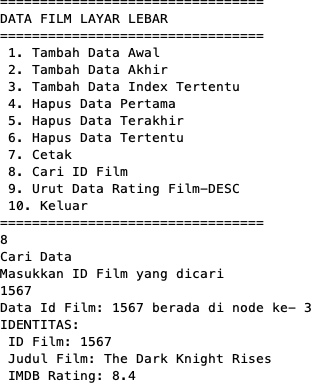
## Contoh Ilustrasi Program

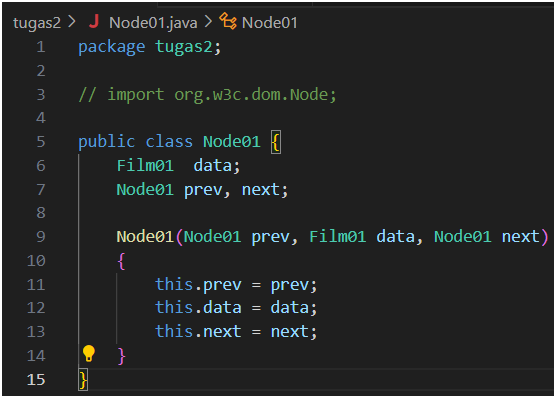
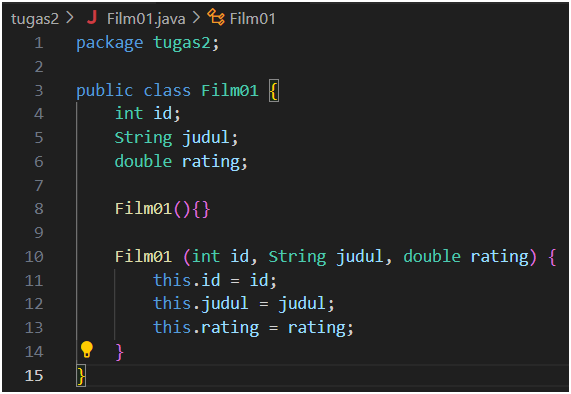
 *Menu Awal dan Penambahan Data*

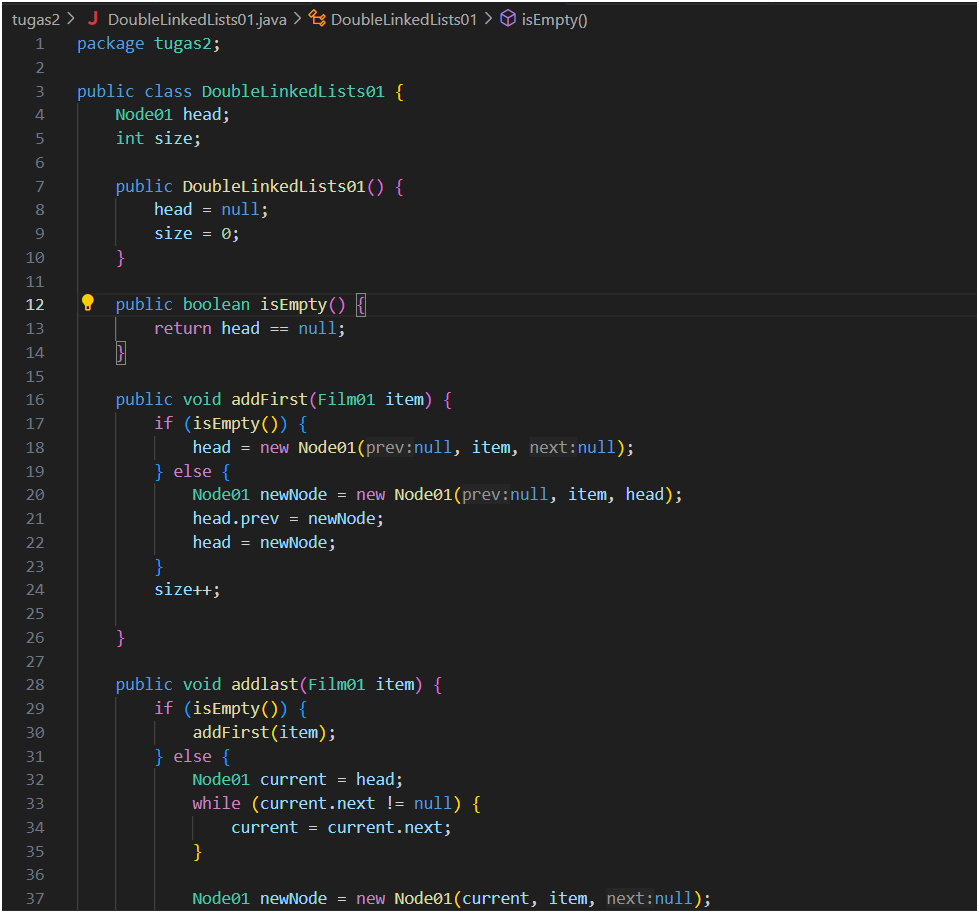
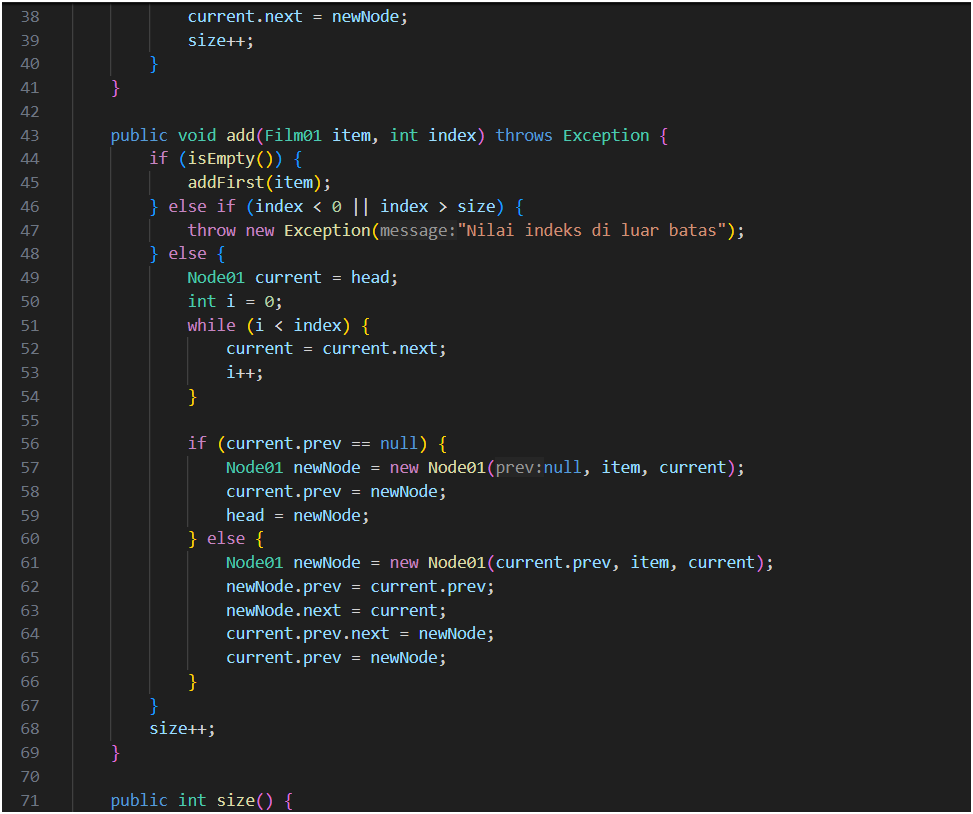
|  |  |
| --- | --- |
|  | Text  Description automatically generated |



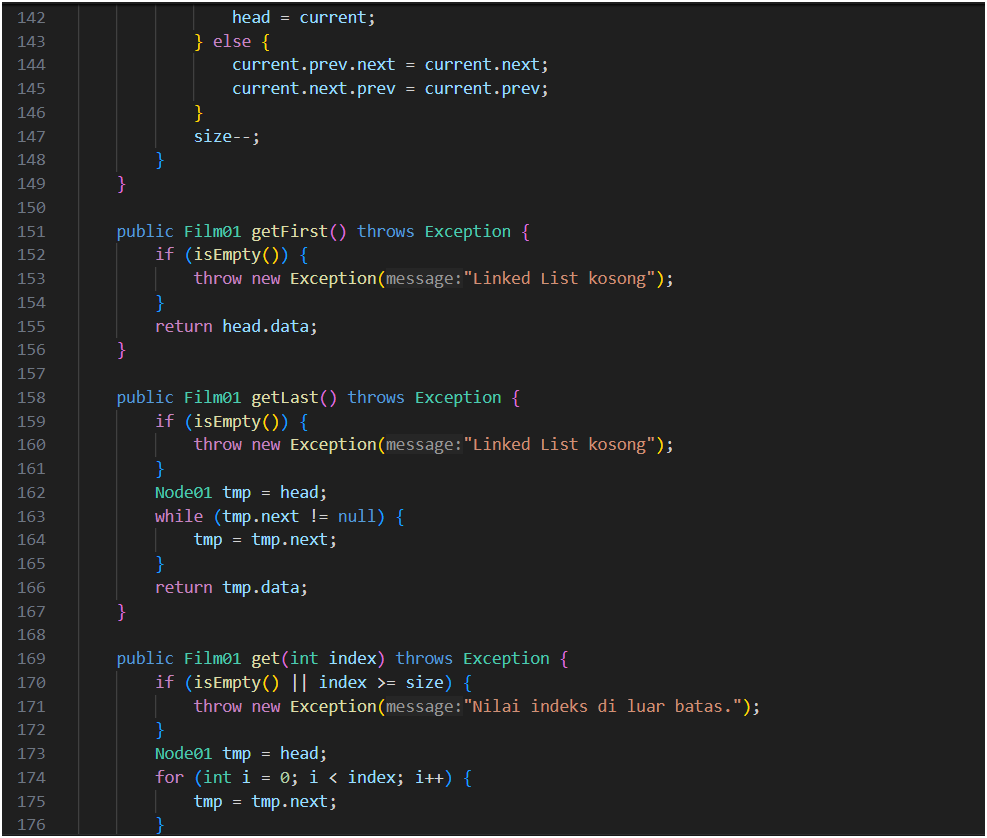
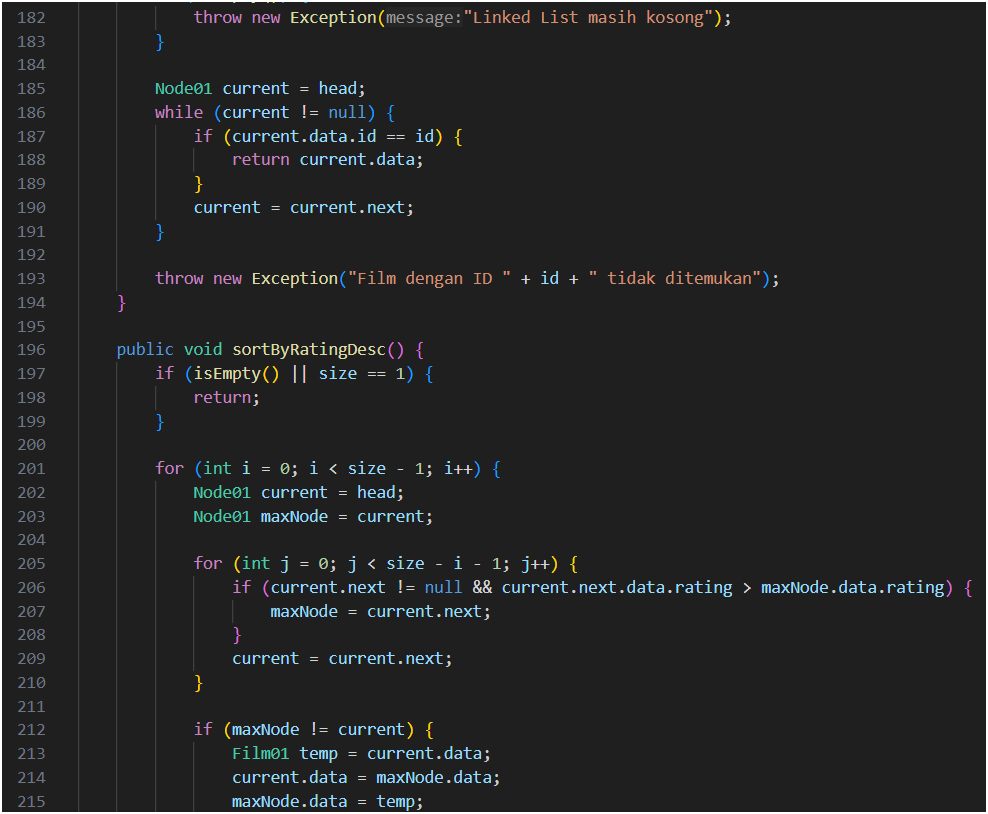
*Cetak Data*

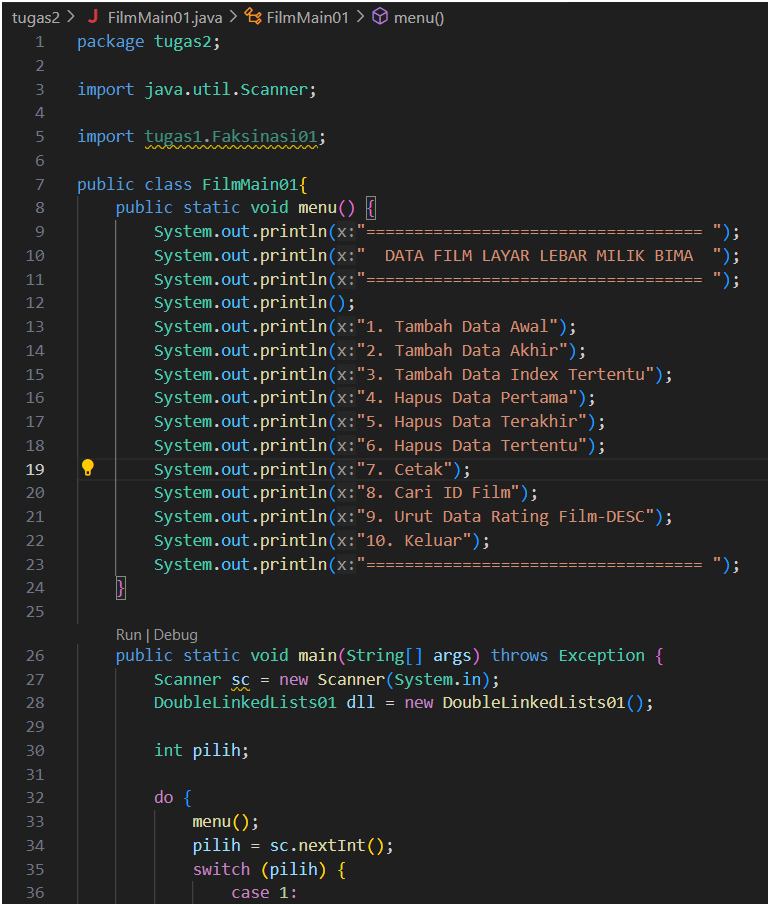
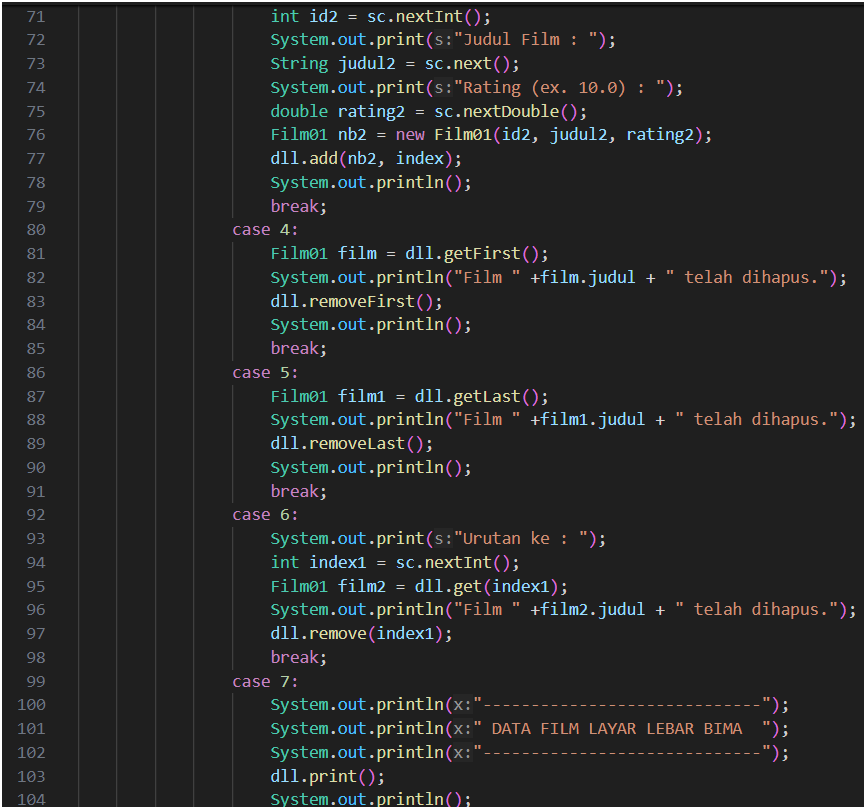
*Cetak Data Pencarian Data*

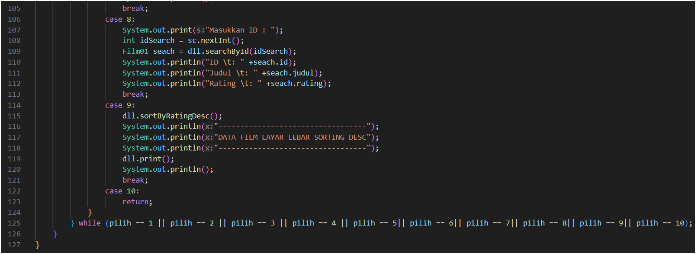
***Class node01 Class film01***

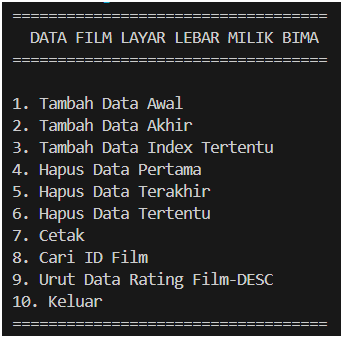
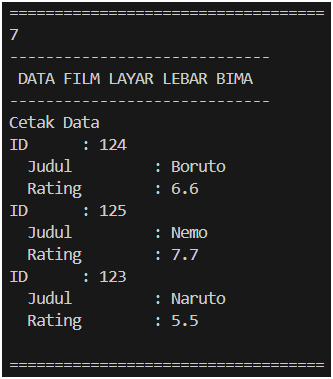
***Class doubleLinkedLists01***

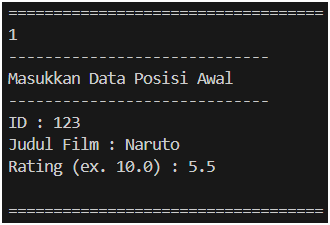


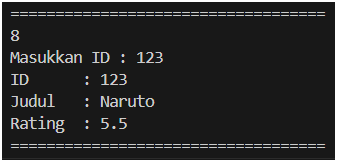


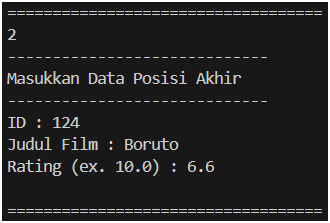
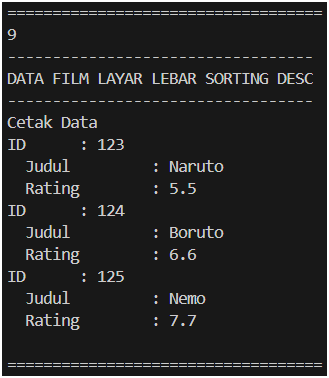
***Class FileMain01***

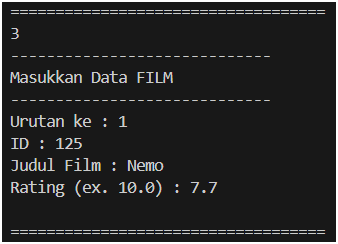
******

***Output Cetak***

***Posisi awal***

****** ***Cari ID***

***Posisi akhir Urut Data Rating***

***Data Index tertentu***