# LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

Dosen Pengajar : Triana Fatmawati, S.T, M.T

# Jobsheet-12 Double Linked List



Nama : Surya Rahmat Fatahillah

NIM : 2341760020

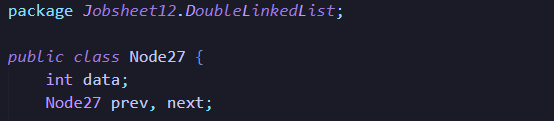
Prodi : Sistem Informasi Bisnis

# JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG 2023/2024

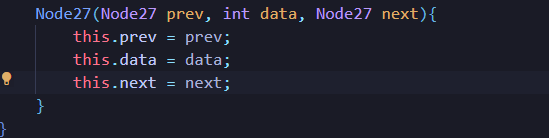
**12.2 Kegiatan Praktikum 1**

**12.2.1 Percobaan 1**

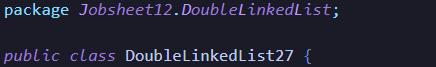
Membuat paket baru dengan nama doublelinkedlists dan class di dalam paket tersebut dengan nama Node, kemudian di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas.



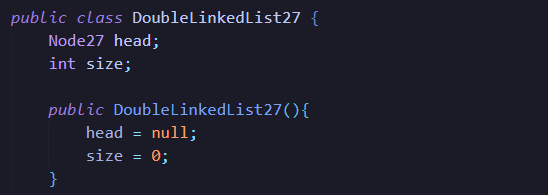
Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.



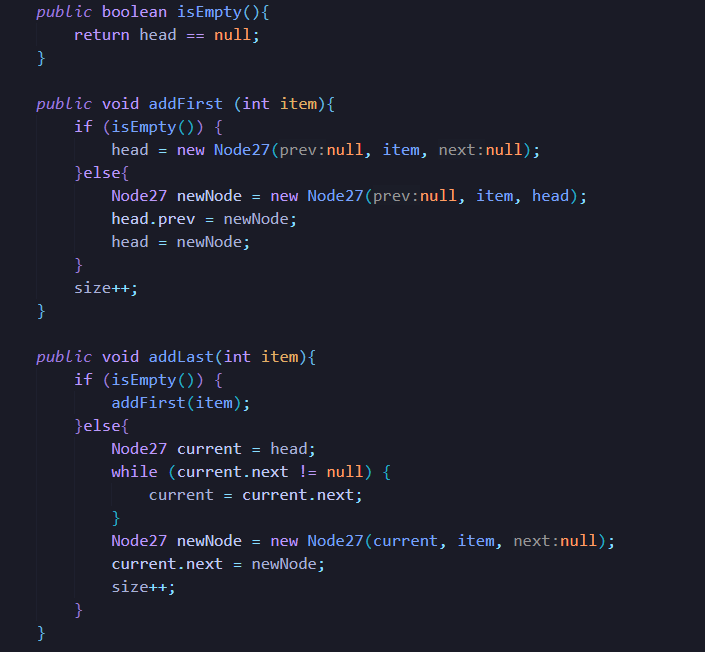
Membuat sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:



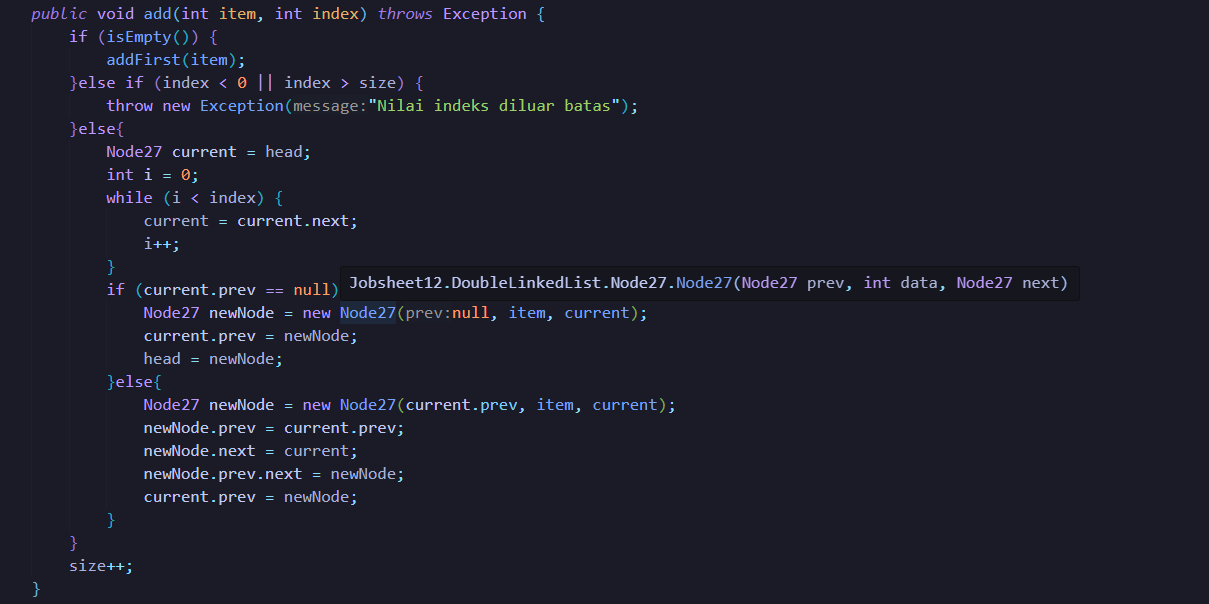
Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas, serta berikan konstruktor pada class DoubleLinkedLists sesuai gambar berikut.



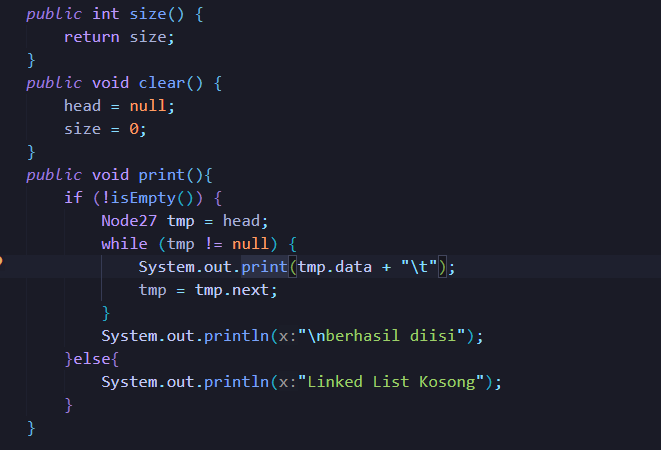
Kemudian buatlah method isEmpty(). Method ini digunakan untuk memastikan kondisi linked list kosong. method addFirst(). Method ini akan menjalankan penambahan data di bagian depan linked list. . Selain itu pembuatan method addLast() akan menambahkan data pada bagian belakang linked list.



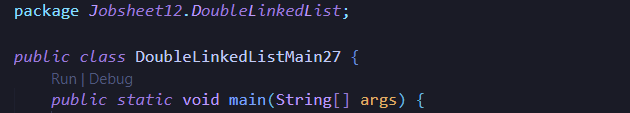
Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method add(int item, int index).



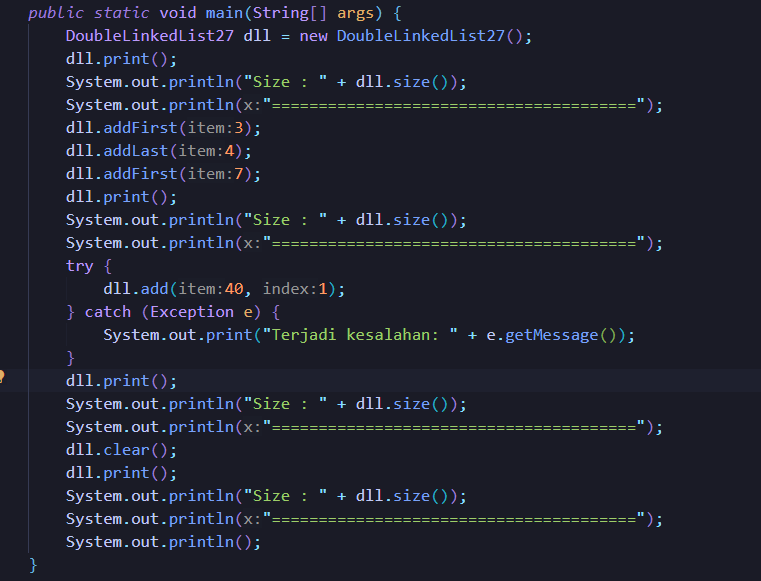
Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis,sehingga dapat dibuat method size() untuk mendapatkan nilai dari size. Selanjutnya dibuat method clear() untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong. Untuk mencetak isi dari linked lists 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.



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

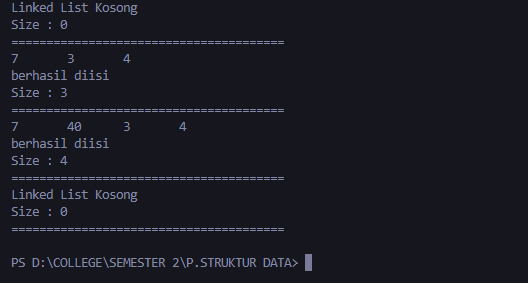


Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.



**12.2.2 Verifikasi Hasil Percobaan**

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.



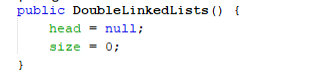
**12.2.3 Pertanyaan Percobaan**

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

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

tersebut?

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



4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev

dianggap sama dengan null?

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

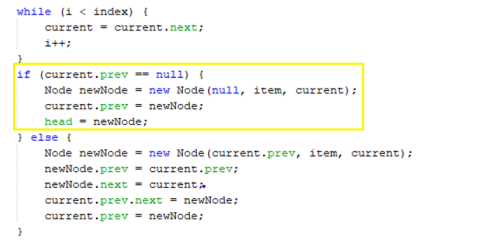
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?

6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan

parameter prev dengan current, dan next dengan null?

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

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



Jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

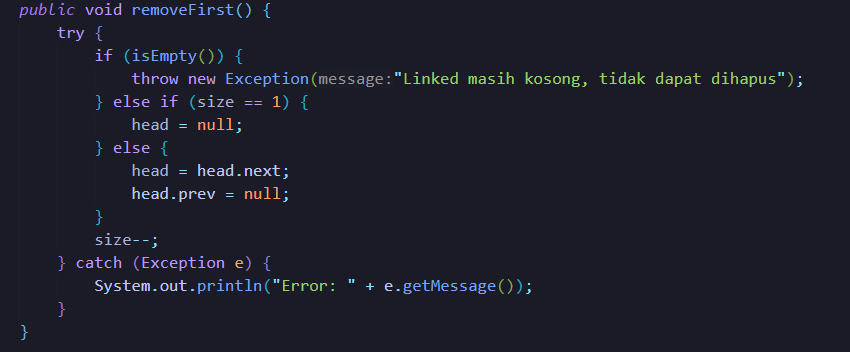
**JAWABAN!**

1. Setiap elemen dalam single linked list disimpan dalam sebuah node yang terdiri dari dua bagian: data dan referensi (atau pointer) ke node berikutnya dalam list. Sedangkan Setiap elemen dalam double linked list juga disimpan dalam sebuah node, namun node ini memiliki dua referensi: satu ke node sebelumnya dan satu ke node berikutnya dalam list. Kemudian Node terakhir dalam single linked list memiliki referensi null, menandakan akhir dari list, sedangkan Node pertama memiliki referensi ke null untuk menandakan awal list, dan node terakhir memiliki referensi ke null untuk menandakan akhir list. Pada single Linked List Traversing (melintasi) dari awal ke akhir list adalah mudah dan efisien. Namun, traversing mundur atau mengakses elemen sebelumnya memerlukan pencarian dari awal list, karena tidak ada referensi langsung ke node sebelumnya, sedangkan Traversing bisa dilakukan mundur (dari akhir ke awal) dengan mudah karena setiap node memiliki referensi ke node sebelumnya.
2. Atribut next dan prev dalam class Node27 digunakan untuk menyimpan referensi ke node berikutnya dan node sebelumnya dalam suatu double linked list. Atribut next adalah referensi ke node berikutnya dalam double linked list. Sedangkan Atribut prev adalah referensi ke node sebelumnya dalam double linked list.
3. Pada konstruktor tersebut, inisialisasi atribut head dan size dilakukan untuk menetapkan status awal dari objek DoubleLinkedList27. Dengan mengatur head ke null, konstruktor menandakan bahwa pada awalnya, linked list belum memiliki node apapun. Dengan mengatur size ke 0, konstruktor menandakan bahwa pada awalnya, linked list tidak memiliki elemen.
4. Dalam method addFirst() dari kelas DoubleLinkedList27, pada saat membuat objek dari konstruktor kelas Node27, parameter prev diinisialisasi dengan null. Ini dilakukan karena ketika menambahkan elemen baru di posisi pertama dalam linked list, elemen tersebut akan menjadi elemen pertama atau kepala (head) dari linked list. Karena elemen baru ini akan menjadi elemen pertama, maka tidak ada elemen sebelumnya di depannya (prev), sehingga nilainya diatur menjadi null. Jadi, pada tahap awal pembuatan linked list, saat hanya ada satu elemen (yang baru ditambahkan di posisi pertama), tidak ada elemen sebelumnya. Oleh karena itu, nilai prev dari elemen pertama diatur menjadi null.
5. Pernyataan head.prev = newNode mengatur pointer prev dari kepala (head) sebelumnya untuk menunjuk ke elemen baru (newNode), sehingga elemen baru tersebut menjadi elemen pertama dalam linked list dan elemen sebelumnya (yang tadinya kepala) menjadi kedua. Dengan demikian, elemen baru akan menjadi kepala baru dan elemen sebelumnya akan menjadi elemen kedua dalam linked list.
6. Parameter prev diatur menjadi current bertujuan agar pointer prev dari node baru menunjuk ke node terakhir saat ini dalam linked list. Hal ini diperlukan untuk menjaga hubungan dua arah antara node-node dalam double linked list. Dengan menetapkan prev ke current, node baru akan terhubung ke node terakhir dalam linked list sebagai node sebelumnya. Parameter next diatur menjadi null: Karena node baru akan menjadi elemen terakhir dalam linked list, tidak akan ada elemen setelahnya. Oleh karena itu, pointer next dari node baru diatur ke null.
7. if (current.prev == null): Kondisi ini mengecek apakah elemen yang ditunjuk oleh current adalah elemen pertama dalam linked list. Jika current.prev (elemen sebelumnya dari current) adalah null, itu berarti current adalah elemen pertama dalam linked list. Jika kondisi di atas terpenuhi, maka kita tahu bahwa elemen yang akan ditambahkan harus diletakkan di posisi pertama dalam linked list. Dalam blok ini, kita membuat node baru (newNode) yang akan menjadi elemen pertama dalam linked list. Karena elemen ini akan menjadi elemen pertama, prev-nya diatur menjadi null, dan next-nya diatur untuk menunjuk ke current, yang merupakan elemen yang sedang menjadi elemen pertama. Setelah membuat newNode, pointer prev dari current diatur untuk menunjuk ke newNode, sehingga current sekarang menjadi elemen kedua dalam linked list. Kemudian, kita memperbarui head untuk menunjuk ke newNode, karena sekarang newNode adalah elemen pertama dalam linked list.

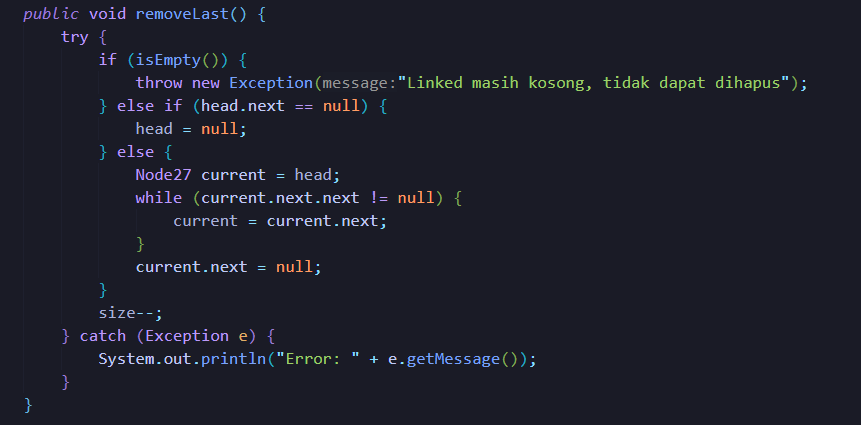
**12.3 Kegiatan Praktikum 2**

**12.3.1 Tahapan Percobaan**

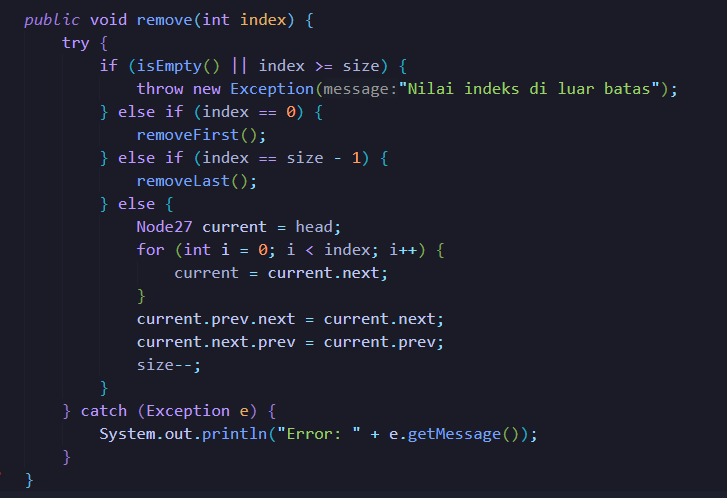
Buatlah method removeFirst() di dalam class DoubleLinkedLists



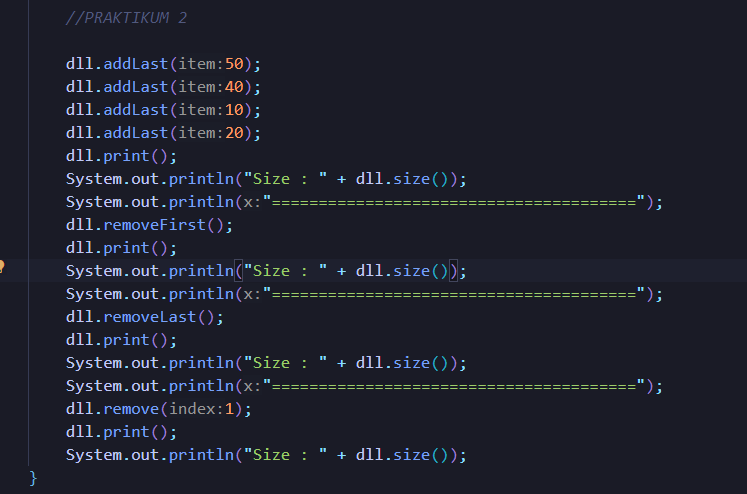
Tambahkan method removeLast() di dalam class DoubleLinkedLists



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

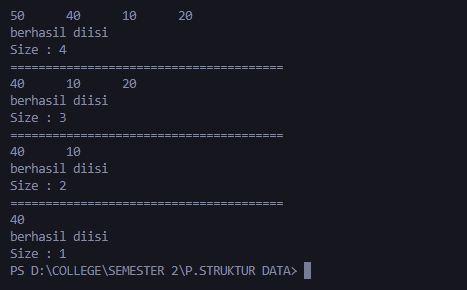


Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada main class.



**12.3.2 Verifikasi Hasil Percobaan**

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.



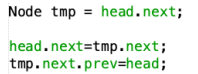
**12.3.3 Pertanyaan Percobaan**

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



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

3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah remove!



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



**JAWABAN!**

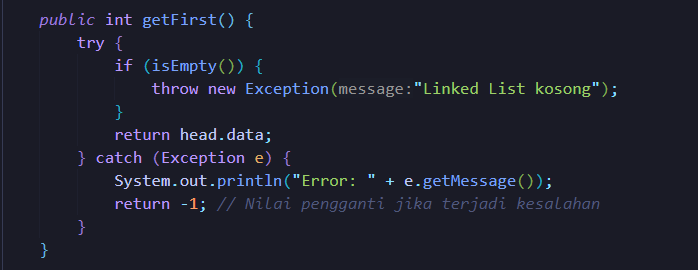
1. Statement head = head.next; di dalam blok else dari metode removeFirst() memiliki maksud untuk memindahkan pointer head ke node berikutnya dalam linked list setelah menghapus elemen pertama. Selanjutnya, statement head.prev = null; bertujuan untuk memutuskan hubungan dengan node sebelumnya, karena node yang sebelumnya adalah elemen pertama, sehingga prev-nya harus diatur menjadi null.
2. Untuk mendeteksi apakah sebuah data berada pada bagian akhir dari linked list saat melakukan operasi removeLast() kita menggunakan loop while untuk menelusuri linked list hingga mencapai elemen kedua terakhir. Ketika current.next.next == null, kita tahu bahwa current adalah elemen kedua terakhir, dan elemen terakhir adalah current.next. Dengan mengatur current.next menjadi null, kita menghapus hubungan dengan elemen terakhir, sehingga elemen tersebut dapat dihapus oleh sistem.
3. Alasan kenapa potongan kode berikut tidak cocok untuk method remove() adalah kode tersebut hanya mengatur pointer next dari head ke tmp.next dan prev dari tmp.next ke head, tanpa memperhatikan node-node sebelum atau setelah tmp. Ini menyebabkan kehilangan hubungan antara node-node sebelum dan setelah tmp.
4. Potongan kode program current.prev.next = current.next; dan current.next.prev = current.prev; digunakan dalam metode remove untuk menghapus node yang ditunjuk oleh current dari linked list. current.prev.next = current.next digunakan untuk mengatur pointer next dari node sebelum current (yang disimpan dalam current.prev) untuk menunjuk ke node setelah current (yang disimpan dalam current.next) dan current.next.prev = current.prev digunakan untuk mengatur pointer prev dari node setelah current (yang disimpan dalam current.next) untuk menunjuk ke node sebelum current (yang disimpan dalam current.prev).

**12.4 Kegiatan Praktikum 3**

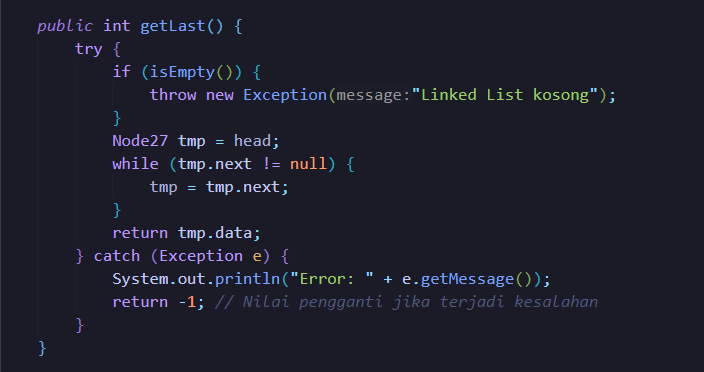
**12.4.1 Tahapan Percobaan**

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

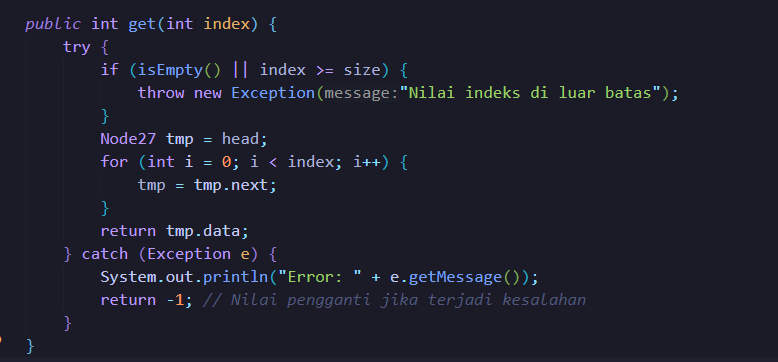
Buatlah method getFirst() di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists



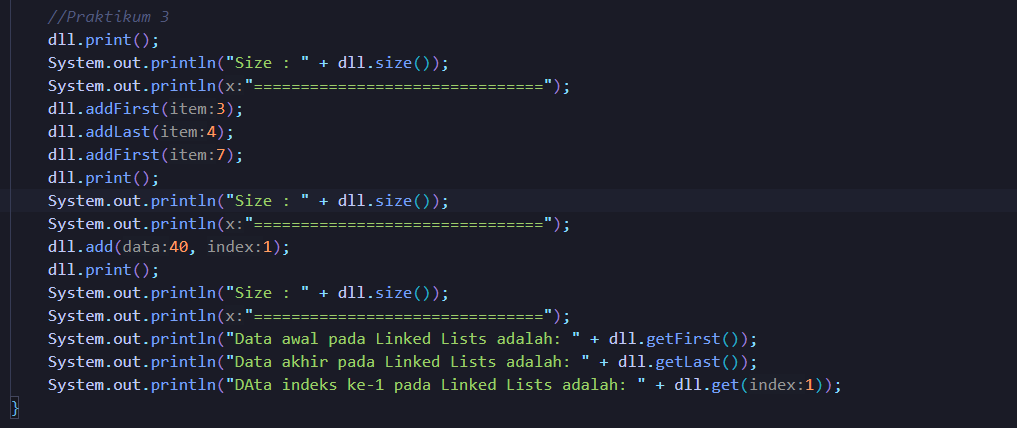
Selanjutnya, buatlah method getLast() untuk mendapat data pada akhir linked lists.



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

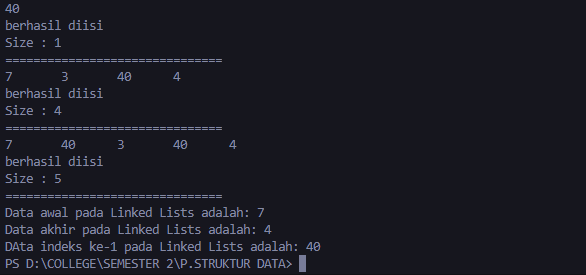


Pada main class tambahkan potongan program berikut dan amati hasilnya!



**12.4.2 Verifikasi Hasil Percobaan**

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini



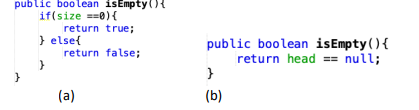
**12.4.3 Pertanyaan Percobaan**

1. Jelaskan method size() pada class DoubleLinkedLists!

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

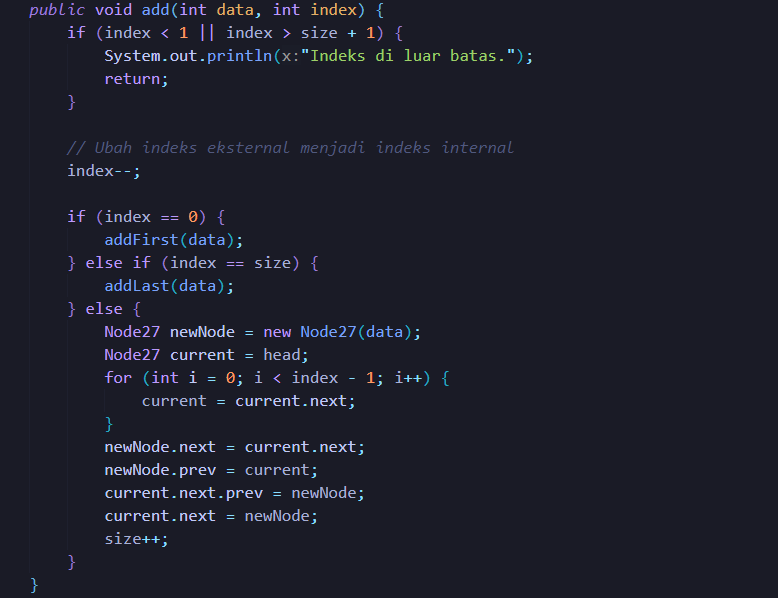
3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!

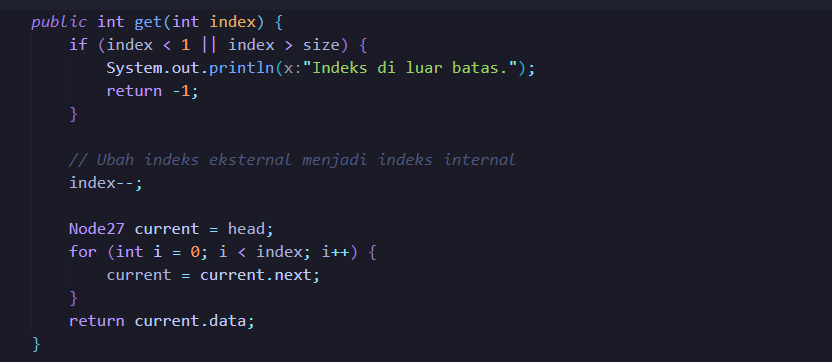
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!



**JAWABAN!**

1. Metode size() pada kelas DoubleLinkedLists bertujuan untuk mengembalikan jumlah elemen yang terdapat dalam linked list pada saat itu.
2. Untuk mengatur indeks pada double linked lists agar dapat dimulai dari indeks ke-1, kita bisa lakukan dengan cara memodifikasi method Get dan Add sehingga mereka menerima indeks eksternal (indeks yang dimulai dari 1) dan mengubahnya menjadi indeks internal (indeks yang dimulai dari 0) sebelum mengakses atau memanipulasi linked list. Berikut modifikasi nya:





1. Berikut beberapa perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!

* Pada Double Linked Lists, bisa menambahkan elemen di berbagai posisi, seperti di awal (head), di akhir (tail), atau di antara dua elemen lainnya. Ini karena setiap node dalam DLL memiliki referensi ke node sebelumnya (prev) dan node berikutnya (next). Sedangkan pada Single Linked Lists, biasanya penambahan elemen hanya bisa dilakukan di awal (head) atau di akhir (tail). Hal ini karena setiap node hanya memiliki referensi ke node berikutnya (next).
* Penambahan elemen di tengah linked list pada Double Linked Lists lebih efisien karena tidak perlu mengubah referensi dari node-node lainnya. Cukup dengan mengatur referensi prev dan next dari node-node yang terkait, sedangkan penambahan elemen di tengah linked list pada Single Linked Lists lebih kompleks karena kita harus menemukan node sebelumnya dan menyesuaikan referensinya untuk menunjuk ke node baru.
* Beberapa operasi penambahan di DLL memiliki tingkat kompleksitas waktu yang konstan (O(1)), seperti penambahan di awal dan akhir (jika kita menyimpan referensi ke tail). Namun, penambahan di tengah memerlukan iterasi melalui linked list, sehingga kompleksitasnya bisa menjadi O(n), dimana n adalah jumlah elemen dalam linked list, sedangkan operasi penambahan di awal linked list pada SLL memiliki tingkat kompleksitas waktu konstan (O(1)), tetapi penambahan di akhir atau di tengah memerlukan iterasi melalui linked list, sehingga kompleksitasnya menjadi O(n).

1. Berikut penjelasan logika pada potogan kode program A

- Kode program ini menggunakan variabel size untuk menentukan apakah linked list kosong atau tidak.

- Jika nilai size sama dengan 0, itu berarti tidak ada elemen dalam linked list, sehingga linked list dianggap kosong, dan kode program mengembalikan true.

- Jika nilai size tidak sama dengan 0 (misalnya lebih besar dari 0), itu berarti ada elemen dalam linked list, sehingga linked list dianggap tidak kosong, dan kode program mengembalikan false.

Berikut penjelasan logika pada potogan kode program B

* Kode program ini menggunakan pengecekan terhadap keberadaan node head untuk menentukan apakah linked list kosong atau tidak.
* Jika head bernilai null, itu berarti tidak ada node dalam linked list, sehingga linked list dianggap kosong.
* Jika head tidak bernilai null, itu berarti ada setidaknya satu node dalam linked list, sehingga linked list dianggap tidak kosong.

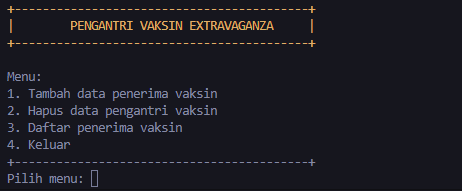
**12.5 Tugas Praktikum**

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 yang telah divaksinasi di menu Hapus Data(2) harus ada).
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

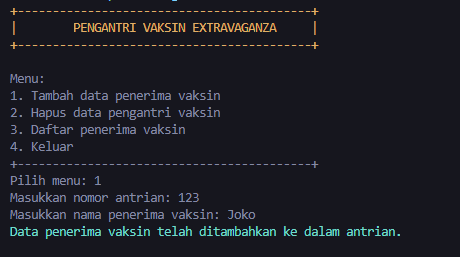
**JAWABAN!**

1. Berikut merupakan program untuk antrian vaksinasi menggunakan queue berbasis double linked list sesuai dengan ilustrasi yang diberikan

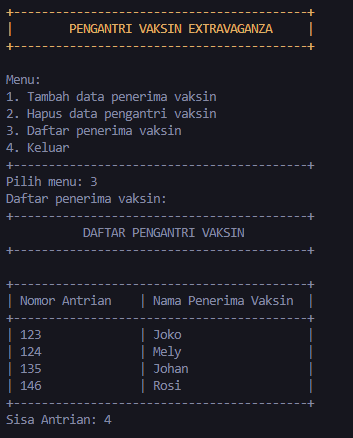
* Menu Awal



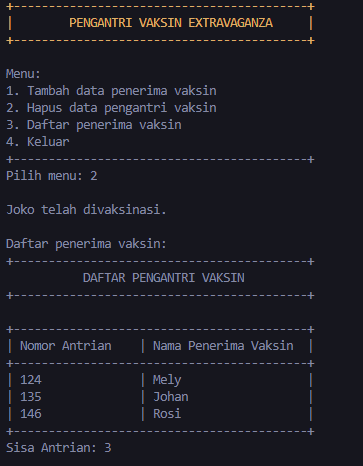
* Fitur Penambahan Data



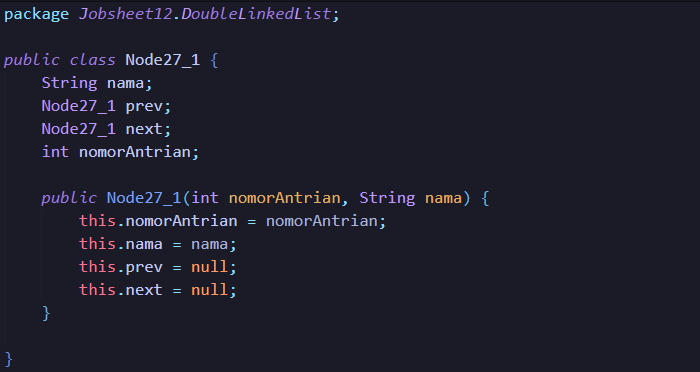
* Fitur Cetak Data



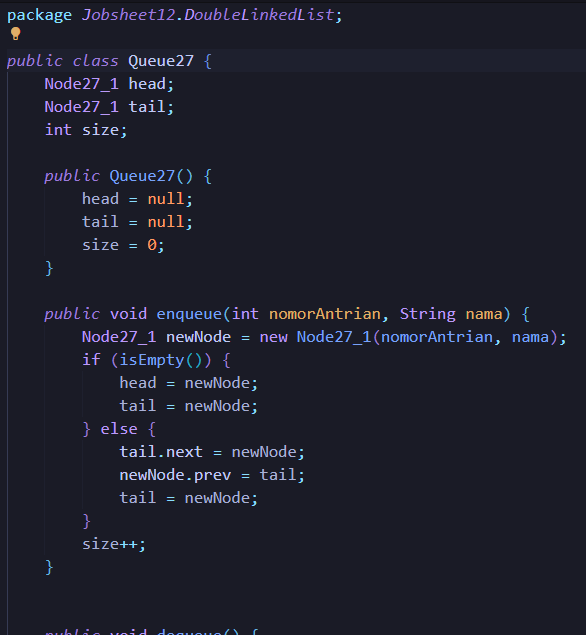
* Fitur Hapus Data



* Class Node



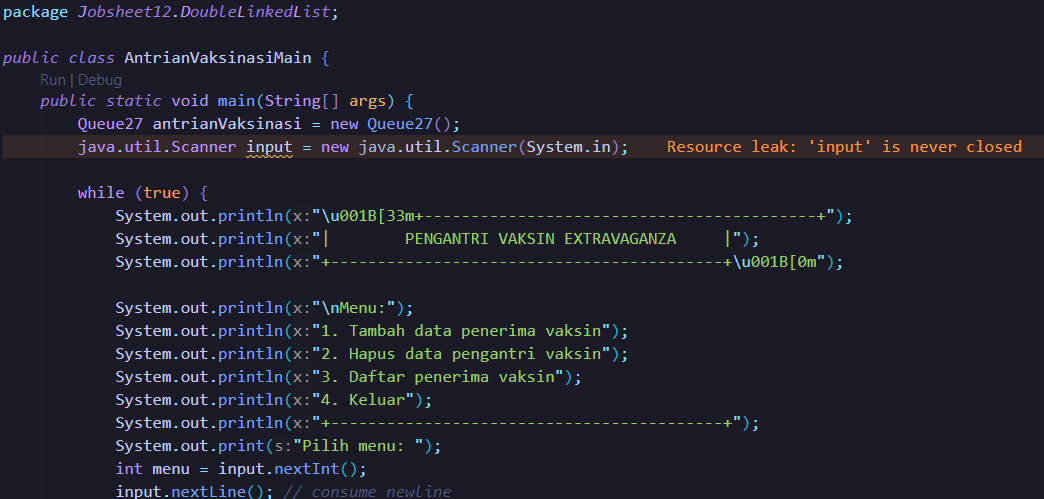
* Class Queue

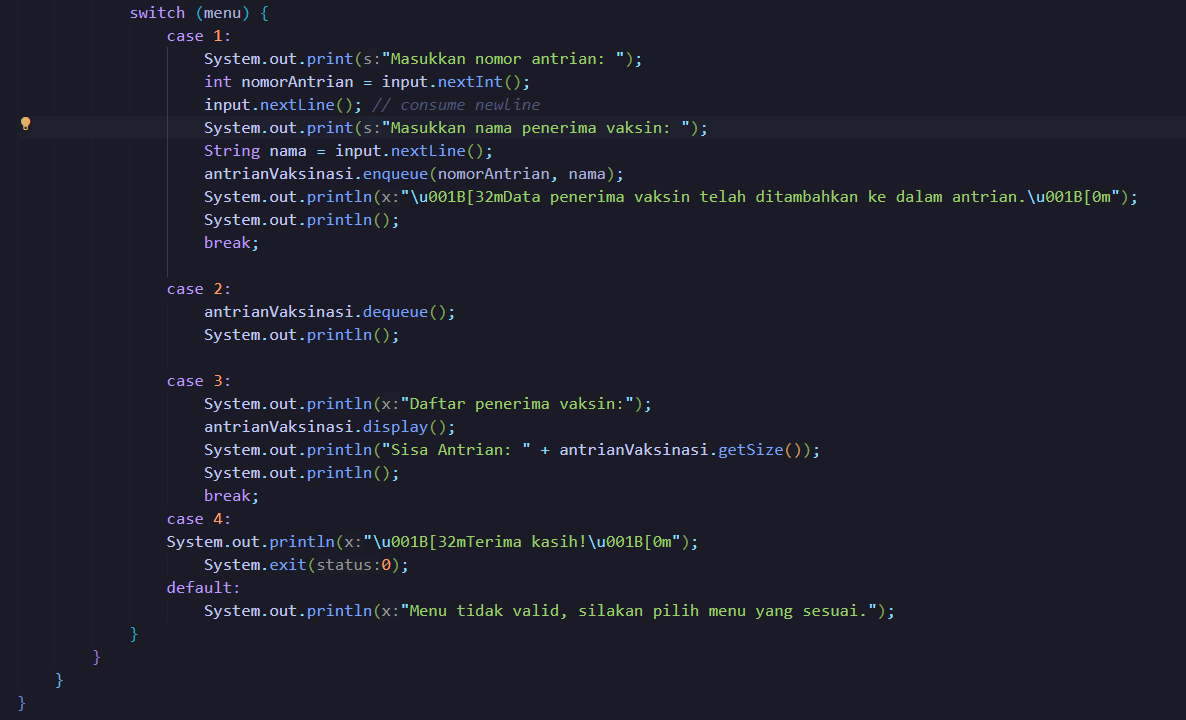






* Class Antrian Vaksinasi Main



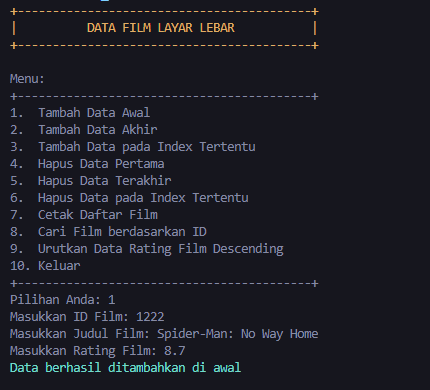


1. Berikut program untuk menampilkan daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists dan sesuai dengan ketentuan.

* Menu Awal



* Tambah Data Awal



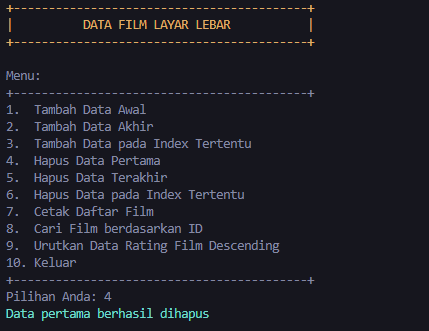
* Tambah Data Akhir

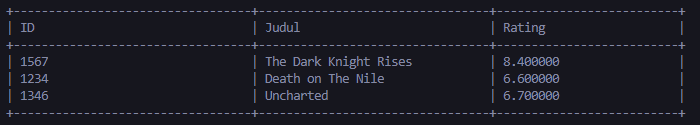


* Tambah Data Pada Index Tertentu



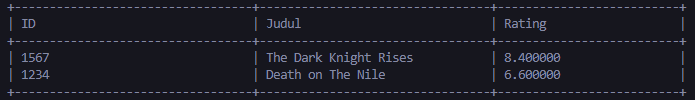
* Hapus Data Pertama





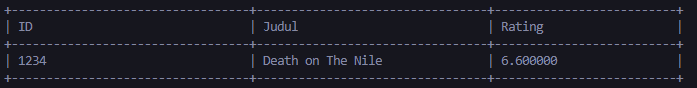
* Hapus Data Terakhir



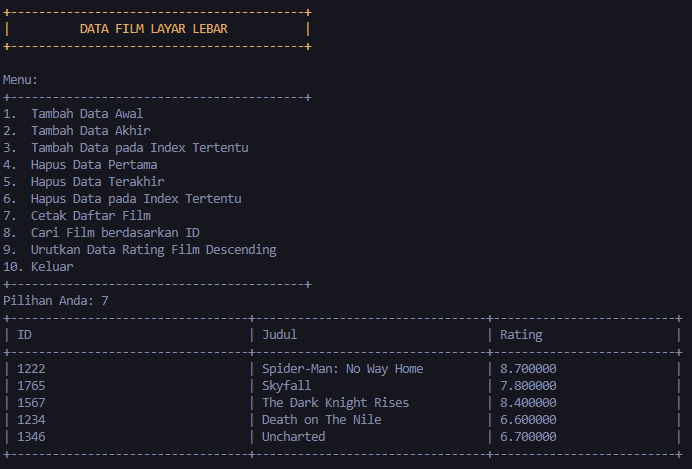


* Hapus Pada Indeks Tertentu





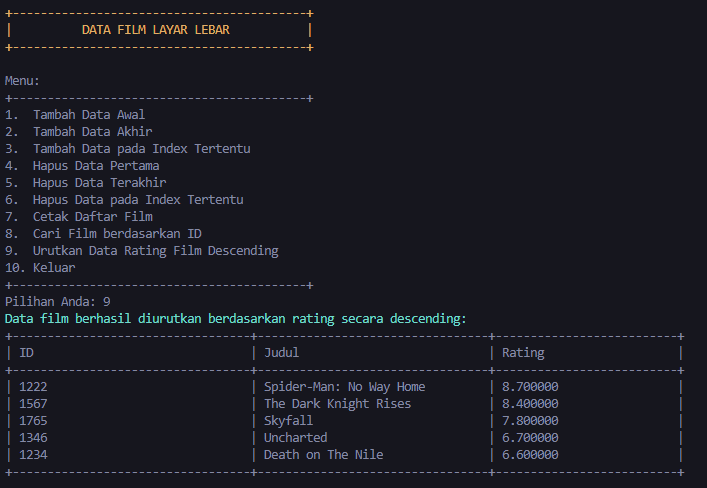
* Cetak Daftar Film



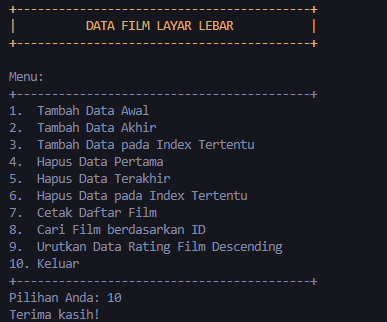
* Cari Film Berdasarkan ID



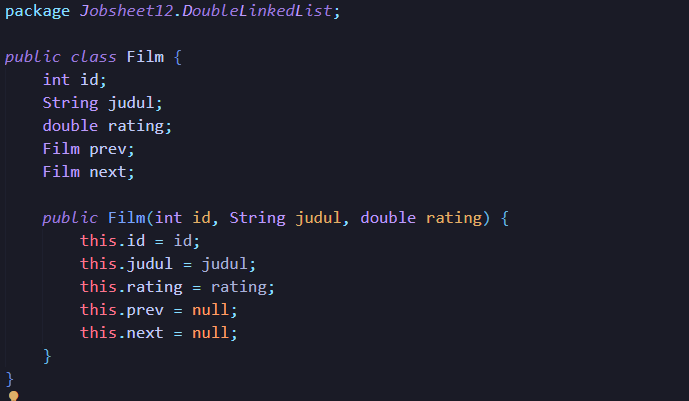
* Urutkan Rating Film Secara Descending



* Keluar

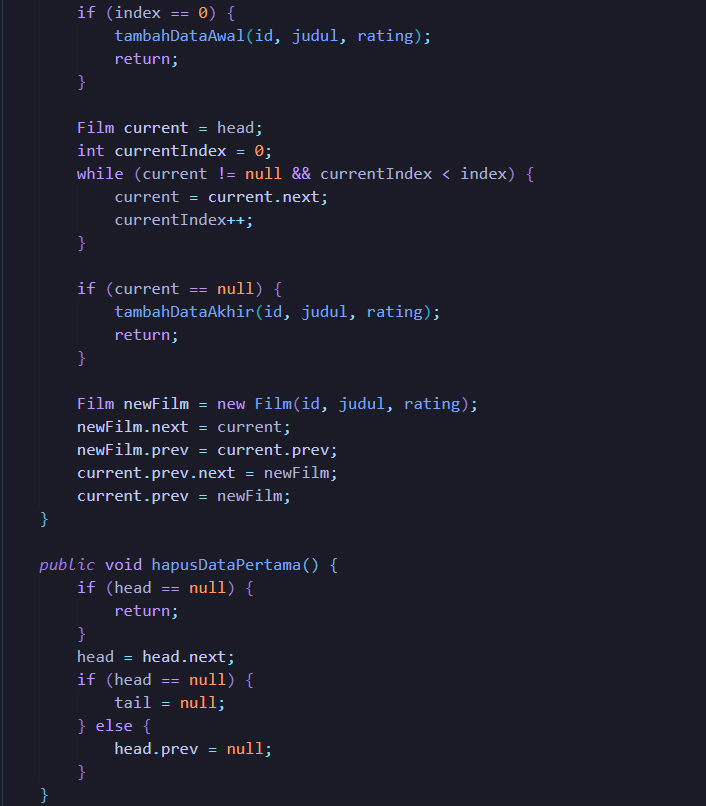


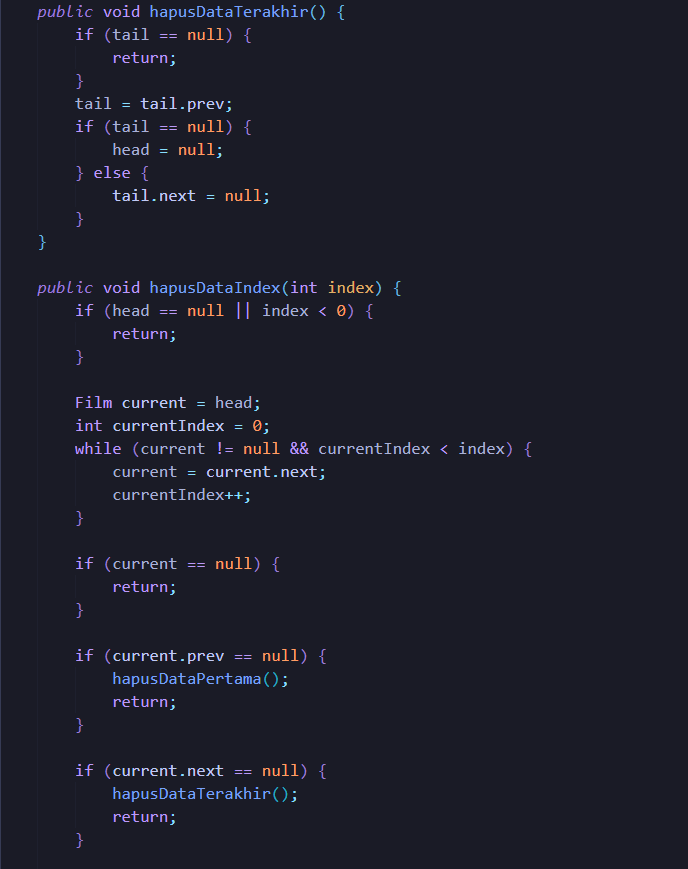
* Class Film

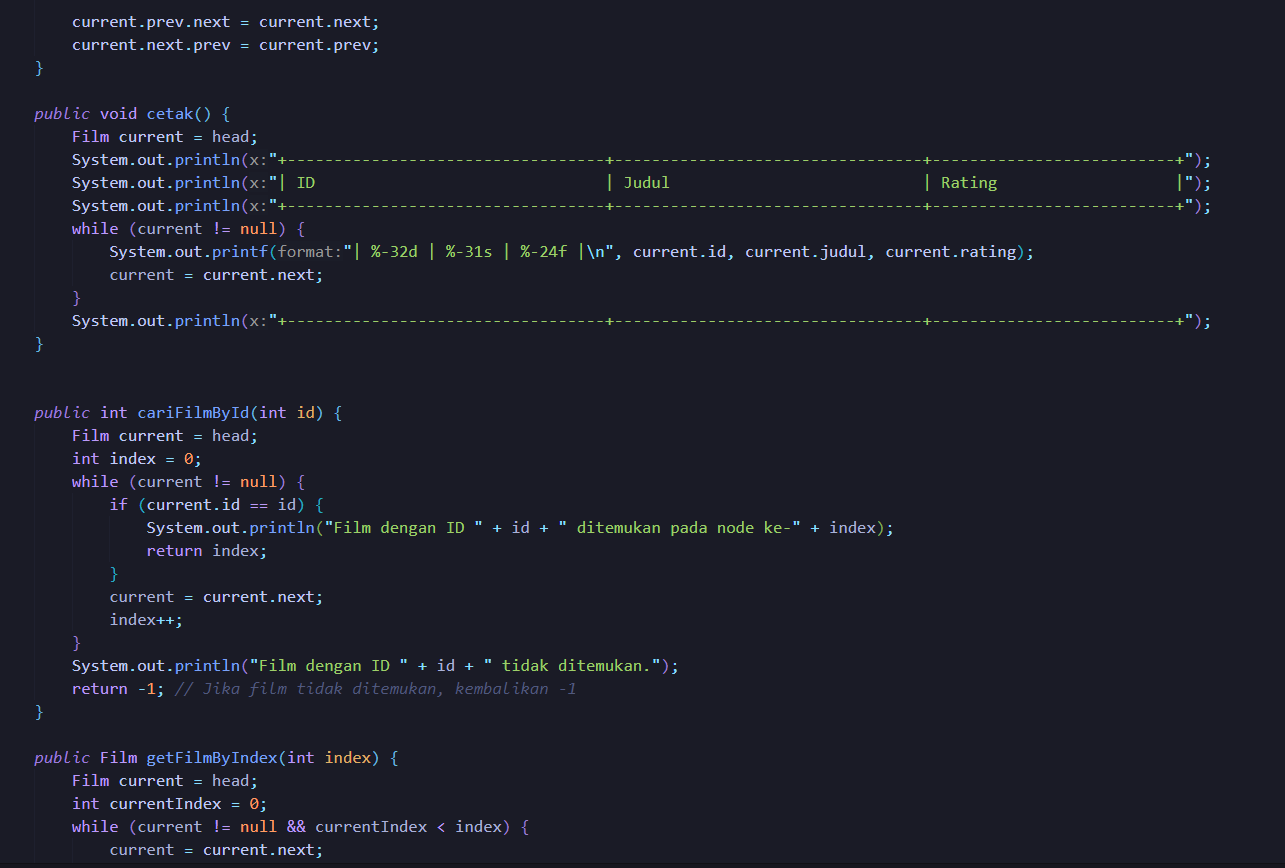


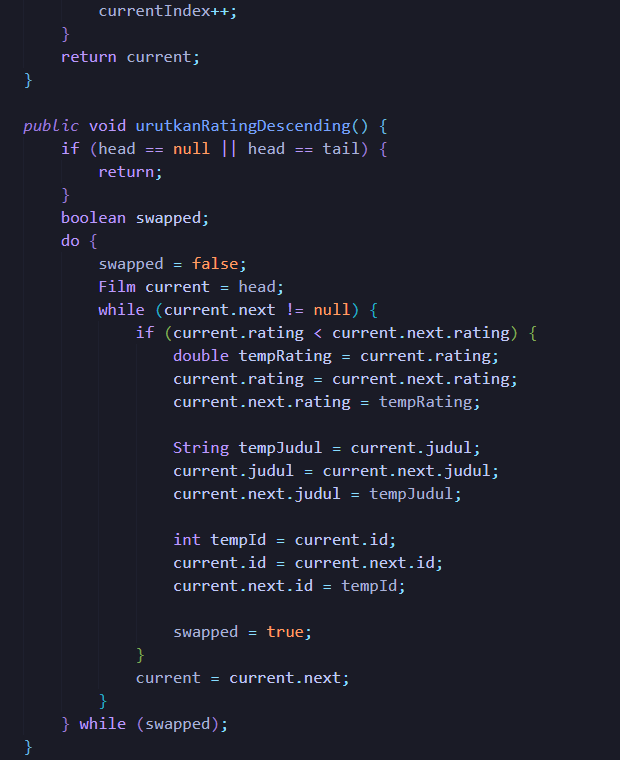
* Class Film List











* Class MainFilm

