JOBSHEET 10

Double Linked Lists

# Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

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

# Kegiatan Praktikum 1

Waktu : 90 Menit

# Percobaan 1

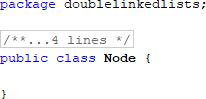
Pada percobaan 1 ini akan dibuat class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan linked list, belakang ataupun indeks tertentu pada linked list).

1. Perhatikan diagram class Node dan class DoublelinkedLists di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.

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

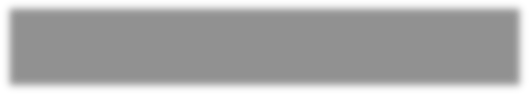
|  |
| --- |
| DoubleLinkedLists |
| head: Node  size : int |
| DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void  add(item: int, index:int): void size(): int  clear(): void  print(): void |

1. Buat paket baru dengan nama **doublelinkedlists**
2. Buat class di dalam paket tersebut dengan nama **Node**



****

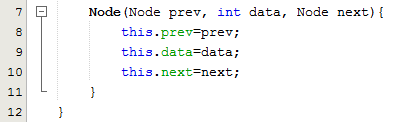
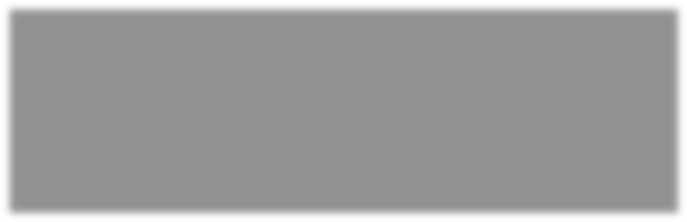
1. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

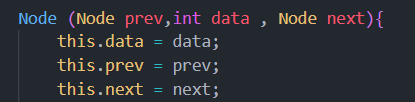




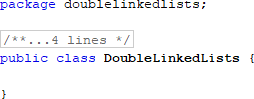
1. Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.

‘



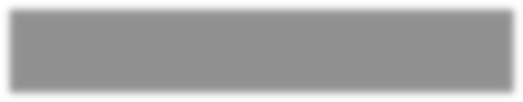


1. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:



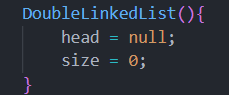


1. Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

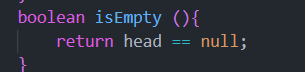


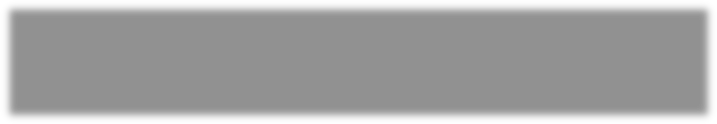


1. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

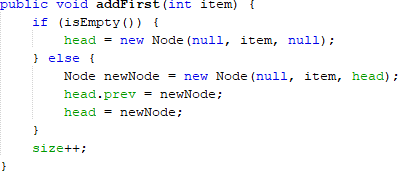


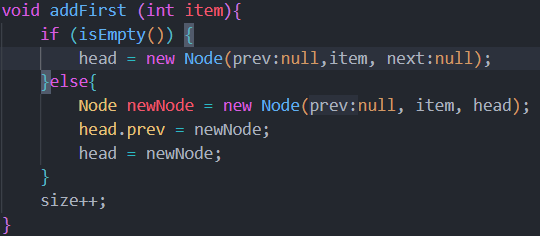
1. Buat method **isEmpty()**. Method ini digunakan untuk memastikan kondisi linked list kosong.

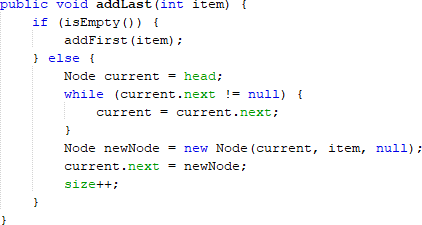


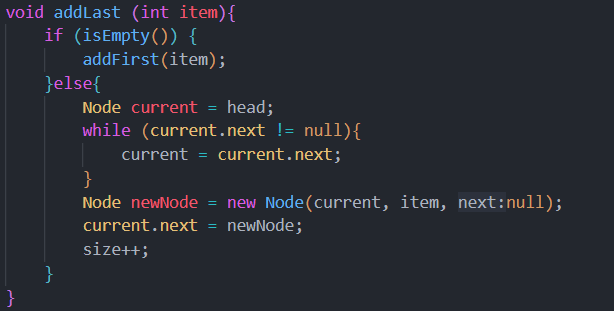


1. Kemudian, buat method **addFirst().** Method ini akan menjalankan penambahan data di bagian depan linked list.

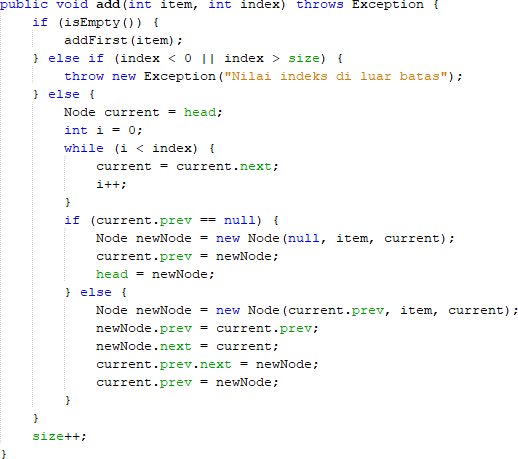


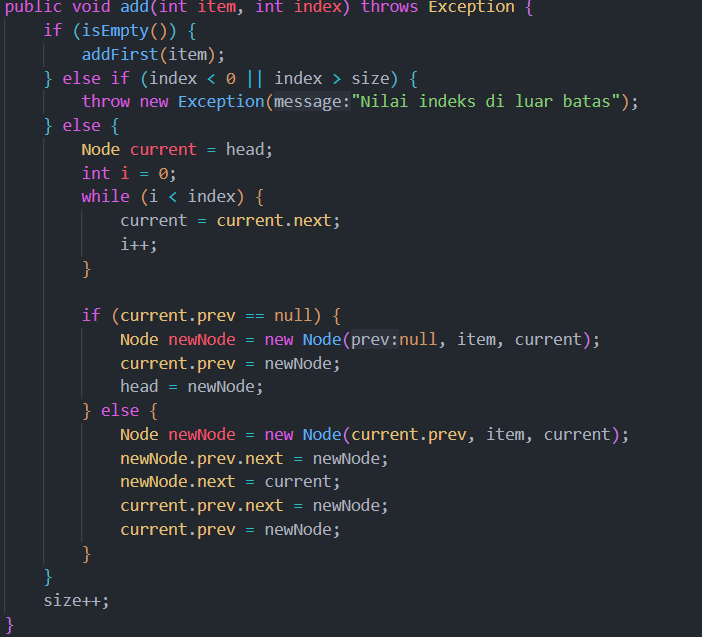


1. Selain itu pembuatan method **addLast()** akan menambahkan data pada bagian belakang linked list.

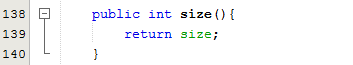
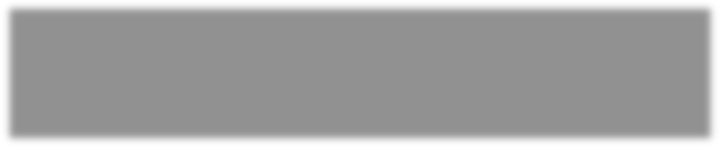


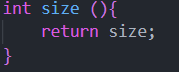
1. Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method **add(int item, int index)**



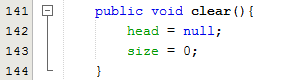
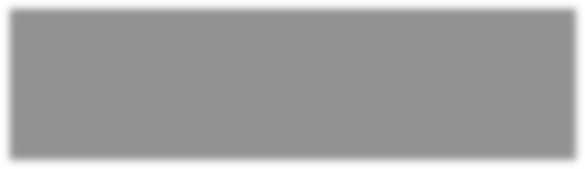


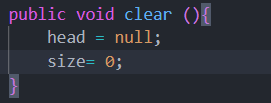
1. Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis, sehingga dapat dibuat method **size()** untuk mendapatkan nilai dari size.

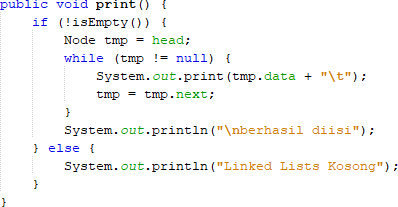


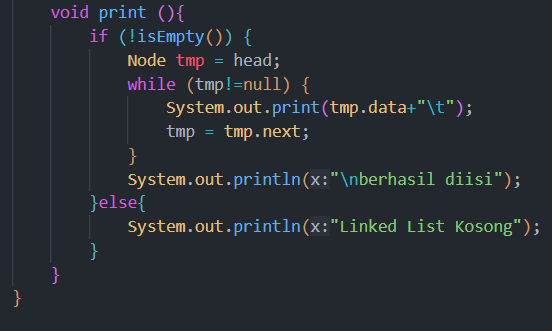


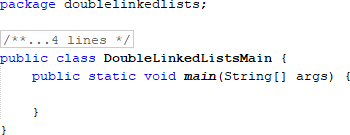
1. Selanjutnya dibuat method **clear()** untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong.

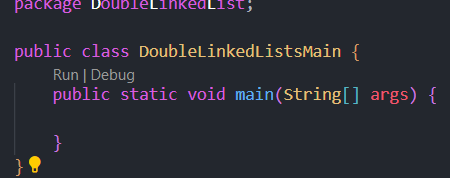




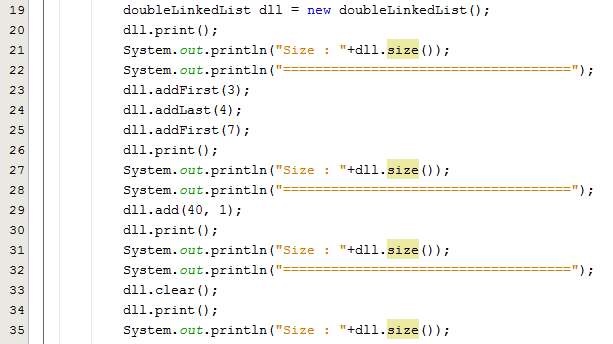
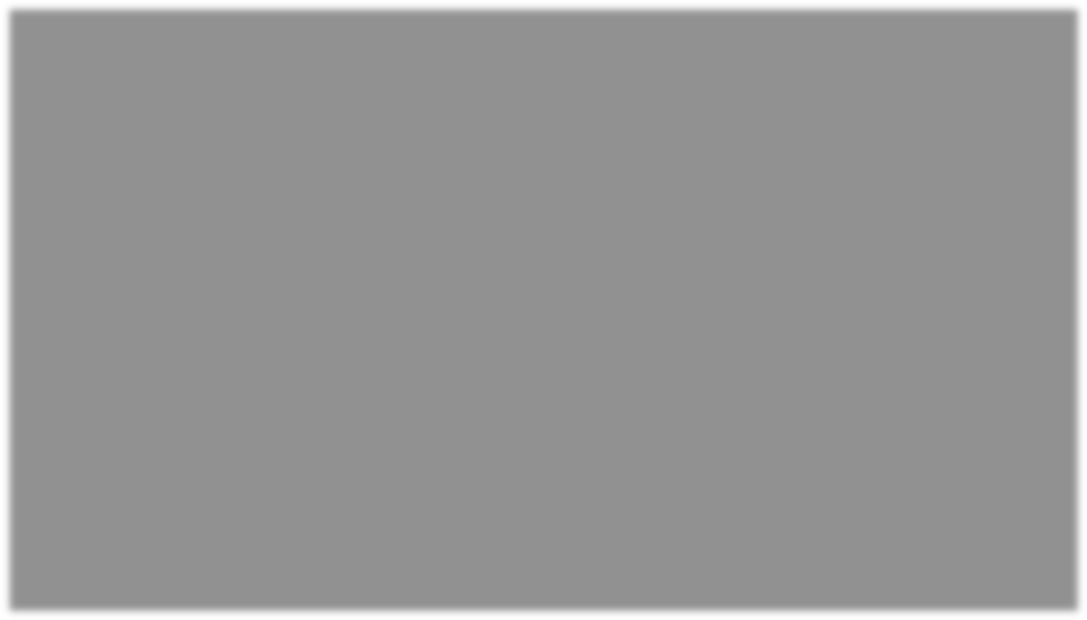
1. Untuk mencetak isi dari linked lists dibuat method **print().** Method ini akan mencetak isi linked lists berapapun size-nya. Jika kosong akan dimunculkan suatu pemberitahuan bahwa linked lists dalam kondisi kosong.



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



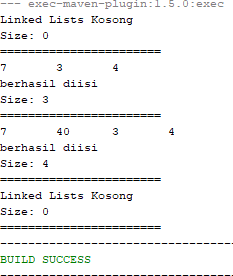
1. Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

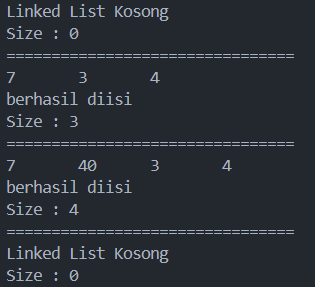




# Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.





# Pertanyaan Percobaan

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

Single linked list setiap node memiliki satu pointer yang menunjuk ke node berikutnya. Sedangkan Double Linked List Setiap node memiliki dua pointer, yaitu satu yang menunjuk ke node sebelumnya (prev) dan satu lagi yang menunjuk ke node berikutnya (next).

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

Atribut next: Menyimpan referensi ke node berikutnya dalam linked list.

Atribut prev: Menyimpan referensi ke node sebelumnya dalam linked list.

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

head = null; : Menginisialisasi linked list kosong dengan tidak ada node awal.

size = 0; : Menginisialisasi ukuran linked list dengan 0 elemen.

1. 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 (addFirst()), tidak ada node sebelumnya untuk node baru pertama, sehingga prev diatur sebagai null.

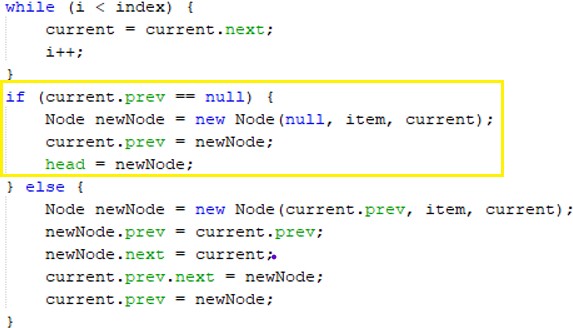
1. Perhatikan pada method **addFirst().** Apakah arti statement head.prev = newNode ?

Statement ini menetapkan bahwa node sebelumnya dari node yang sebelumnya adalah newNode. Dalam kasus ini, head sebelumnya tidak ada atau null, jadi head.prev disetel ke newNode.

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**);

Ini berarti node baru (newNode) ditambahkan di akhir linked list. prev=current mengatur node sebelumnya dari newNode menjadi node yang sekarang menjadi tail. next=null menandakan bahwa newNode adalah node terakhir, sehingga tidak ada node berikutnya setelahnya.

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

jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

Kode ini mengecek apakah current adalah node pertama dalam linked list. Jika iya (current.prev == null), artinya current adalah head dari linked list. Selanjutnya, kode tersebut membuat newNode sebagai node baru yang ditempatkan sebelum current, dan kemudian menetapkan current.prev menjadi newNode. Terakhir, head diupdate menjadi newNode, karena newNode sekarang adalah head baru dari linked list.

# Kegiatan Praktikum 2

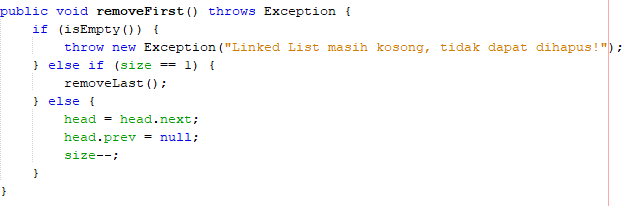
Waktu : 60 Menit

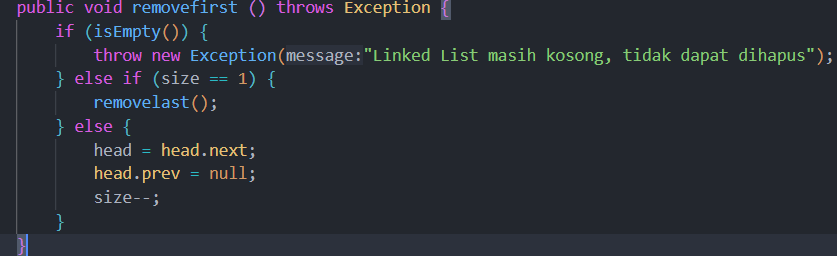
# Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists. Method tambahan tersebut akan ditambahkan sesuai pada diagram class berikut ini.

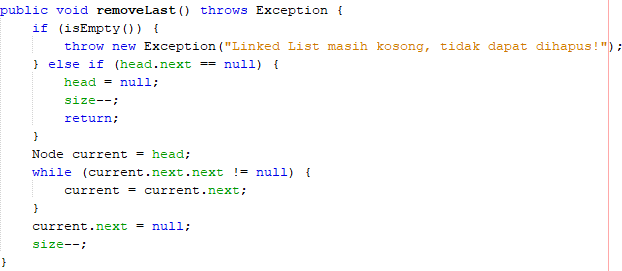
|  |
| --- |
| DoubleLinkedLists |
| head: Node  size : int |
| DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void  add(item: int, index:int): void size(): int  clear(): void print(): void  **removeFirst(): void**  **removeLast(): void remove(index:int):void** |

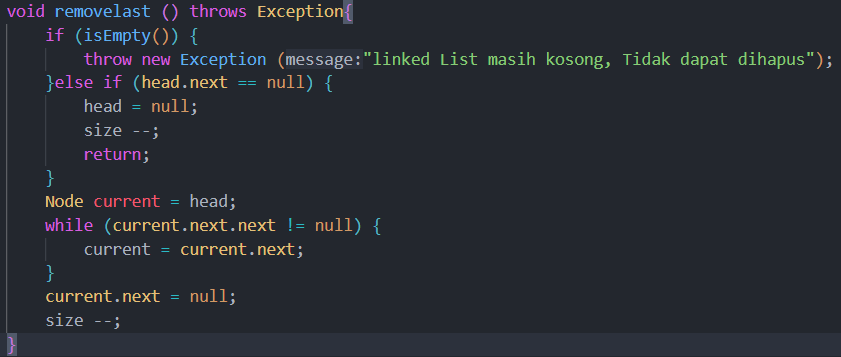
* + - 1. Buatlah method **removeFirst()** di dalam class **DoubleLinkedLists.**

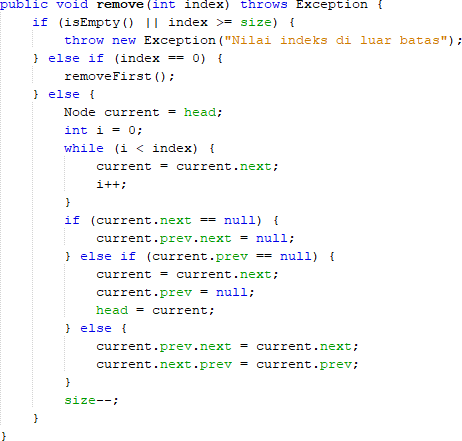


****

* + - 1. Tambahkan method **removeLast()** di dalam class **DoubleLinkedLists.**

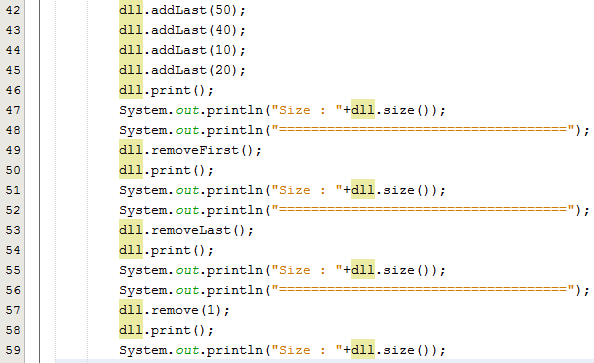
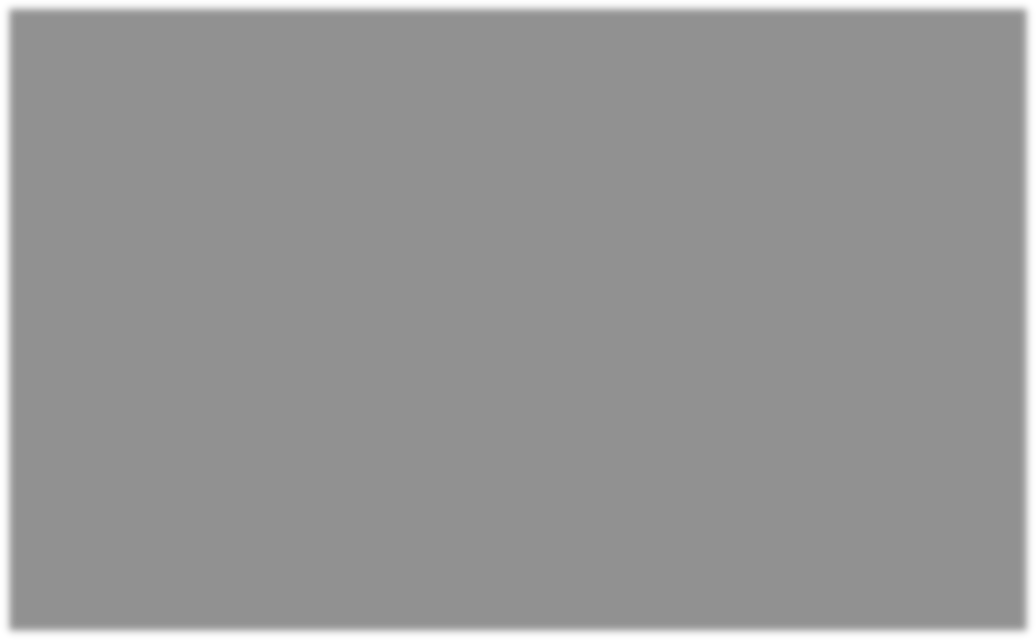




* + - 1. Tambahkan pula method **remove(int index)** pada class **DoubleLinkedLists** dan amati hasilnya.

****

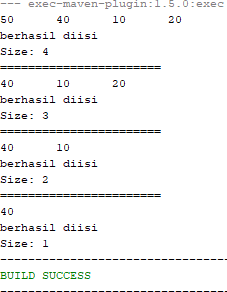
* + - 1. Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada **main class.**

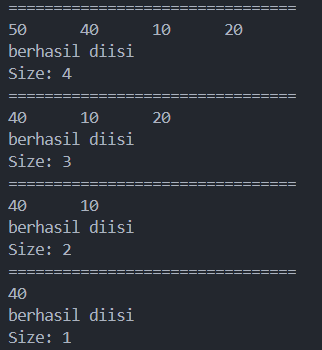




# Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.





# Pertanyaan Percobaan

* + - 1. Apakah maksud statement berikut pada method **removeFirst()**? head = head.next;

head.prev = null;

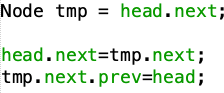
head = head.next;: Statement ini menggeser head ke node berikutnya. Node yang sebelumnya menjadi head akan dilewati dan dihapus dari linked list.

head.prev = null;: Setelah head bergeser ke node berikutnya, statement ini mengatur pointer prev dari node baru yang menjadi head ke null, karena sekarang node ini adalah node pertama dalam linked list dan tidak memiliki node sebelumnya.

* + - 1. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method **removeLast()**?

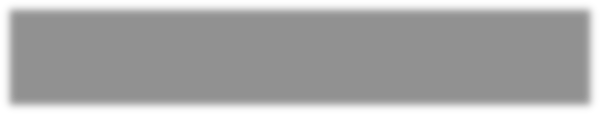
Kita harus mengiterasi melalui linked list hingga mencapai node terakhir. Biasanya, node terakhir adalah node yang memiliki next bernilai null

* + - 1. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah **remove**!



tidak cocok untuk perintah remove karena tidak memeriksa kondisi jika daftar kosong atau hanya memiliki satu elemen. Selain itu, kode tersebut dapat menyebabkan NullPointerException jika tmp.next adalah null.

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



Fungsi dari kedua statement ini adalah untuk menghapus node current dari linked list dengan menghubungkan node sebelum current langsung ke node setelah current, sehingga node current tidak lagi terhubung dalam linked list.

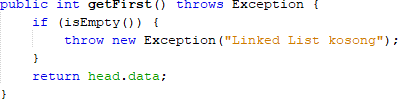
# Kegiatan Praktikum 3

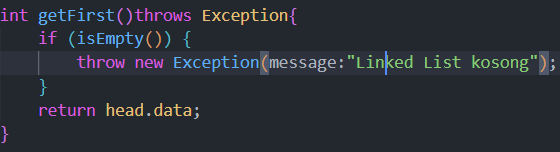
Waktu : 50 Menit

# Tahapan Percobaan

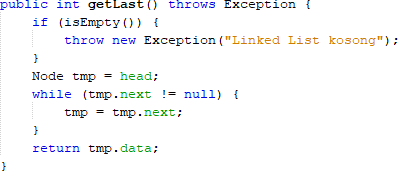
Pada praktikum 3 ini dilakukan uji coba untuk mengambil data pada linked list dalam 3 kondisi, yaitu mengambil data paling awal, paling akhir dan data pada indeks tertentu dalam linked list. Method mengambil data dinamakan dengan **get**. Ada 3 method get yang dibuat pada praktikum ini sesuai dengan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

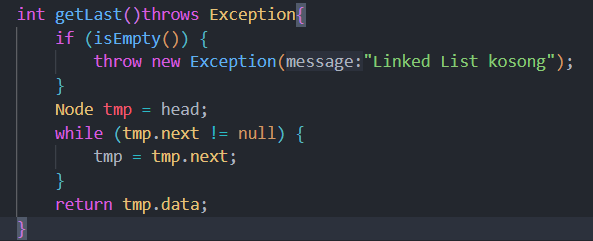
|  |
| --- |
| DoubleLinkedLists |
| head: Node  size : int |
| DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void  add(item: int, index:int): void size(): int  clear(): void print(): void removeFirst(): void removeLast(): void  remove(index:int):void  **getFirst(): int getLast() : int get(index:int): int** |

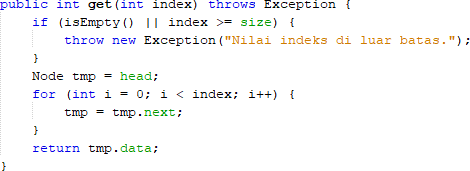
* + - 1. Buatlah method **getFirst()** di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists.

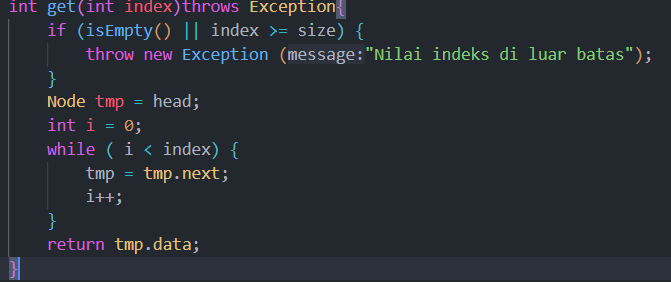


* + - 1. Selanjutnya, buatlah method **getLast()** untuk mendapat data pada akhir linked lists.

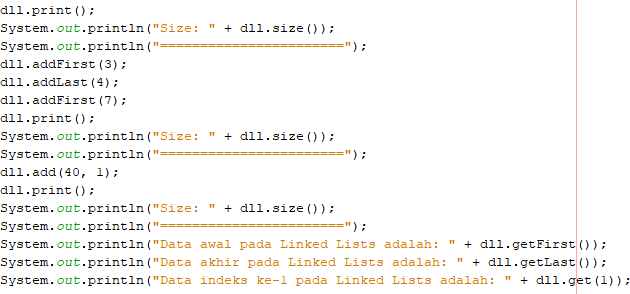


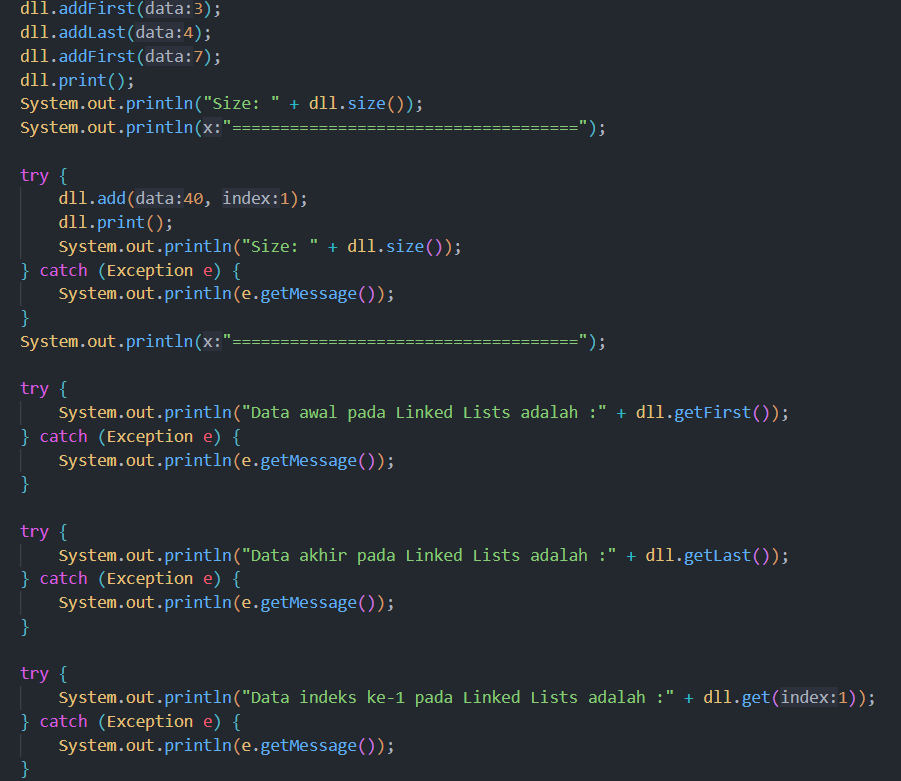


* + - 1. Method **get(int index)** dibuat untuk mendapatkan data pada indeks tertentu



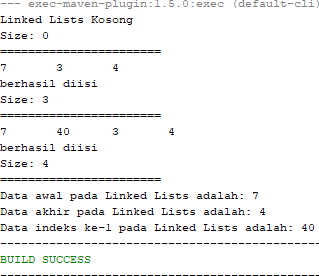
* + - 1. Pada main class tambahkan potongan program berikut dan amati hasilnya!

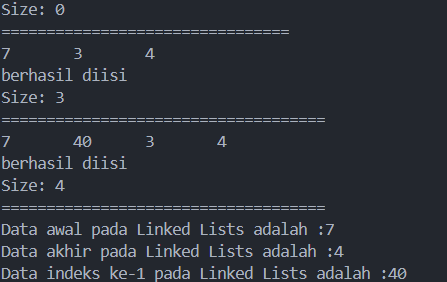




# Verifikasi Hasil Percobaan

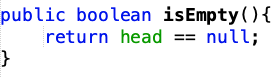
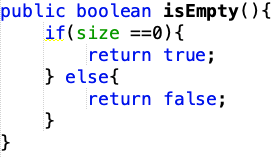
Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.





# 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!



* + - * 1. (b)

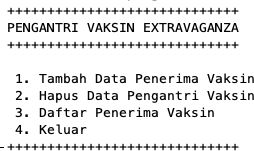
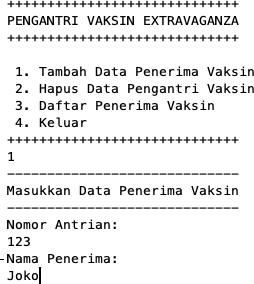
# Tugas Praktikum

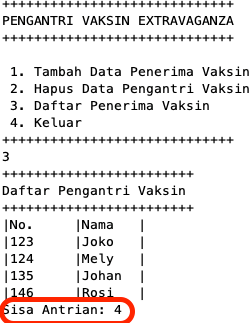
Waktu : 100 Menit

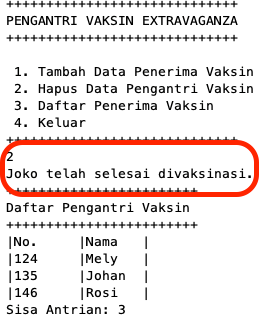
* + 1. Buat program antrian vaksinasi menggunakan queue berbasis double linked list sesuai ilustrasi dan menu di bawah ini! **(counter jumlah antrian tersisa di menu cetak(3) dan data orang yang telah divaksinasi di menu Hapus Data(2) harus ada)**

# Contoh Ilustrasi Program

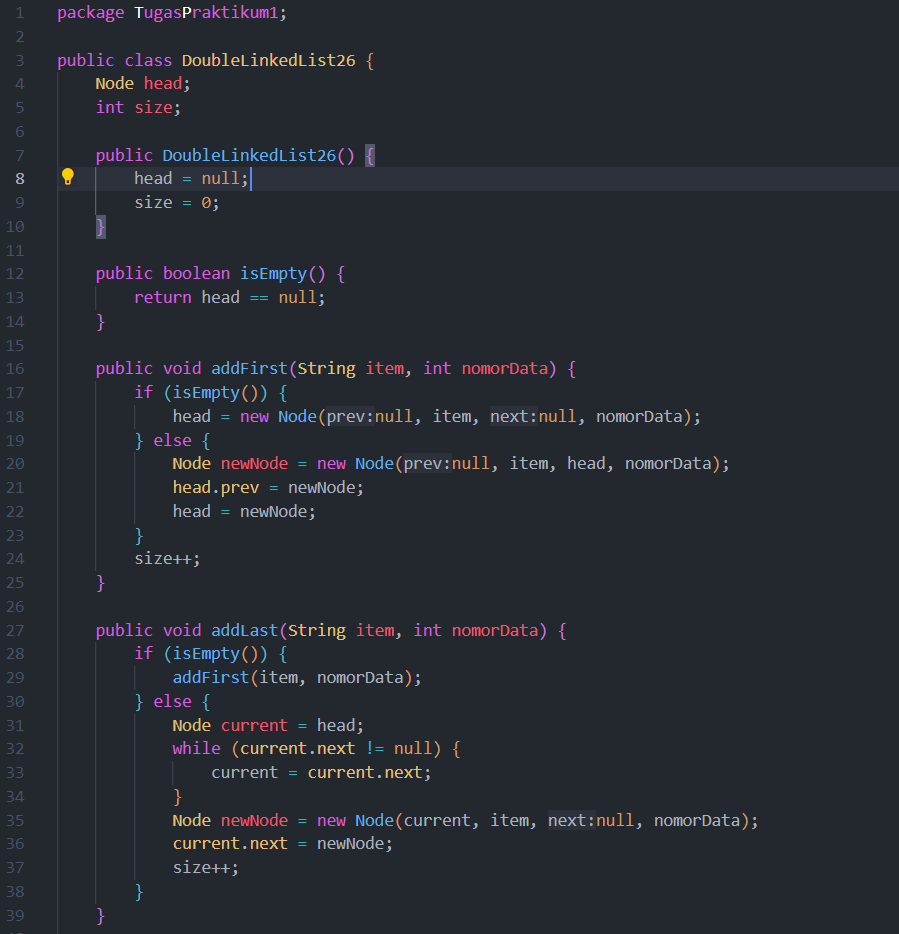
*Menu Awal dan Penambahan Data*

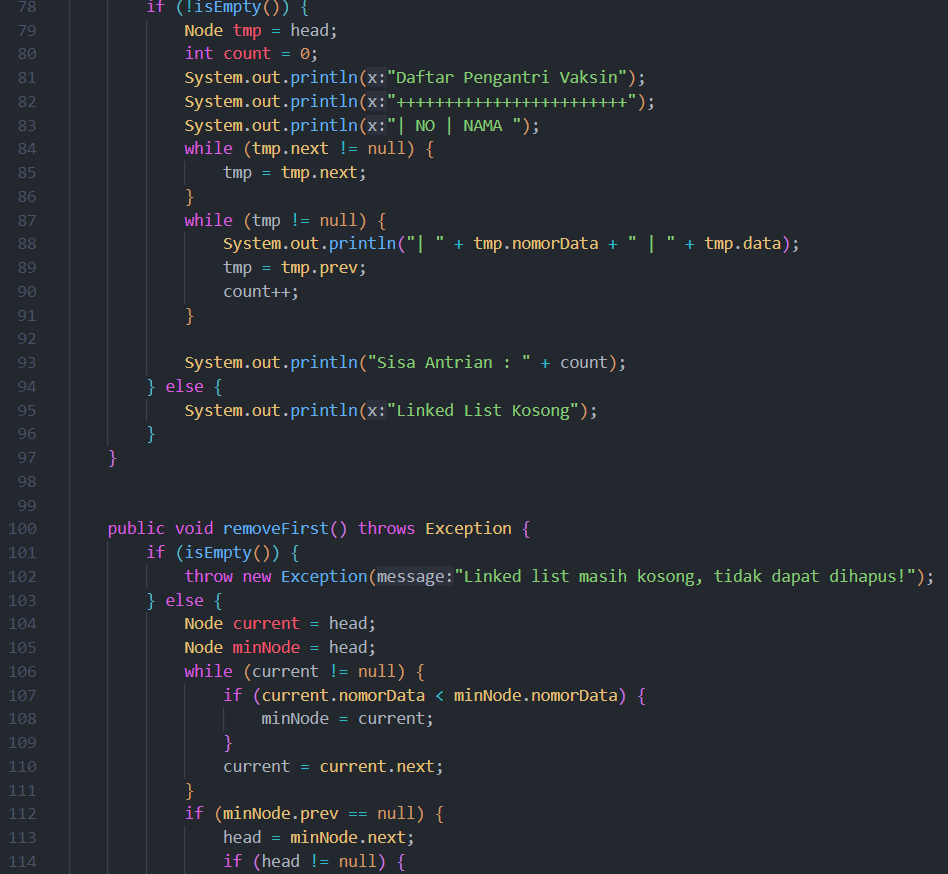
*Cetak Data (****Komponen di area merah harus ada****)*

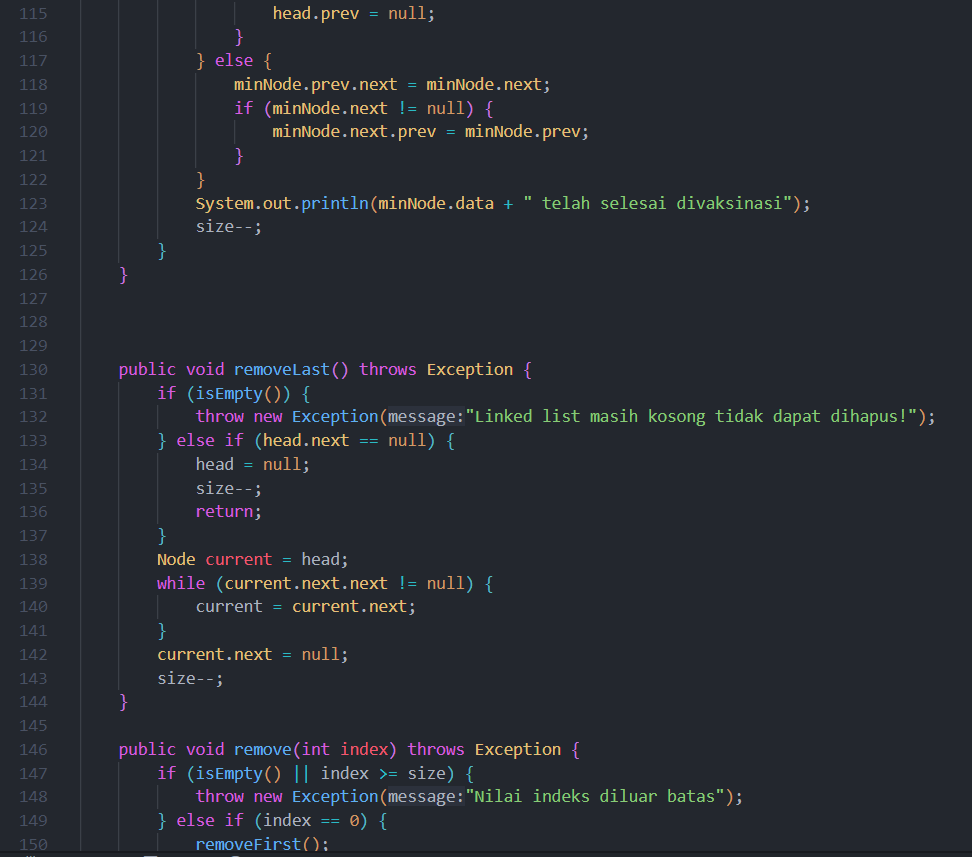
*Hapus Data (****Komponen di area merah harus ada****)*

DoubleLinkedList26:



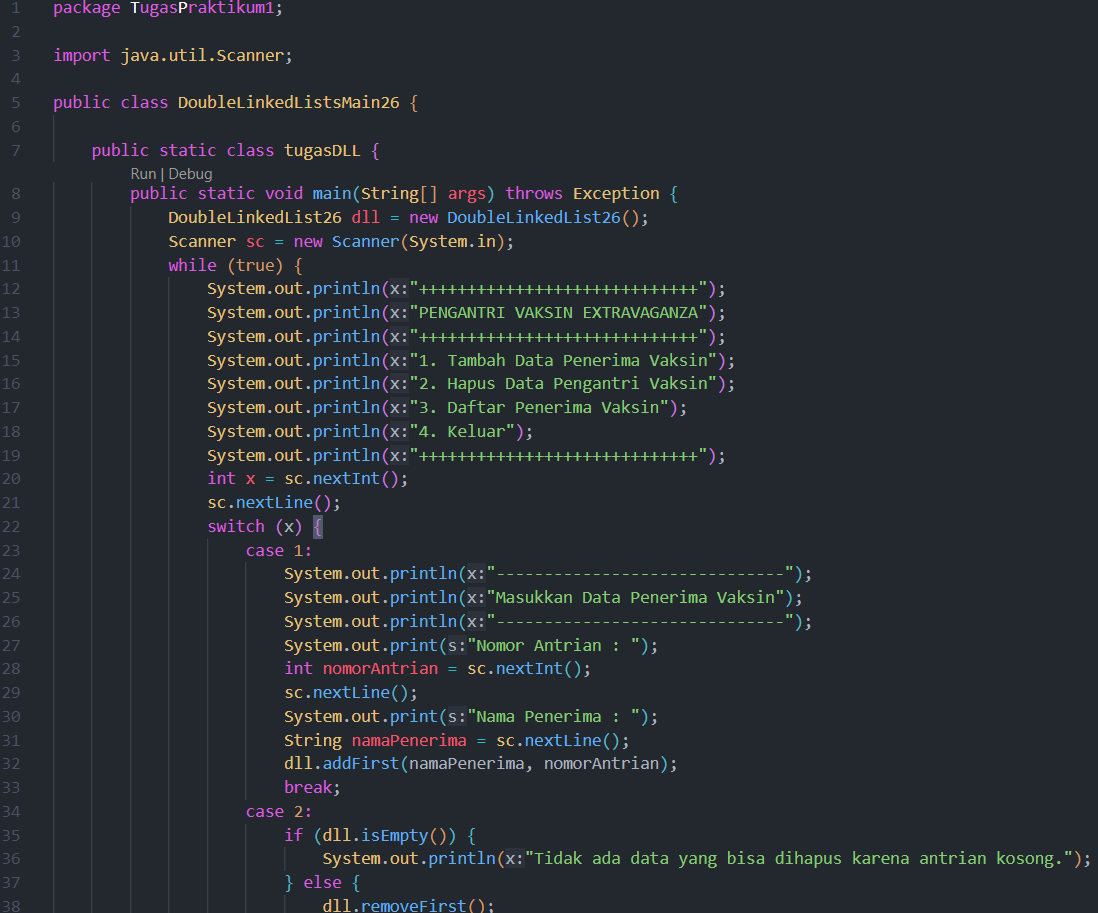


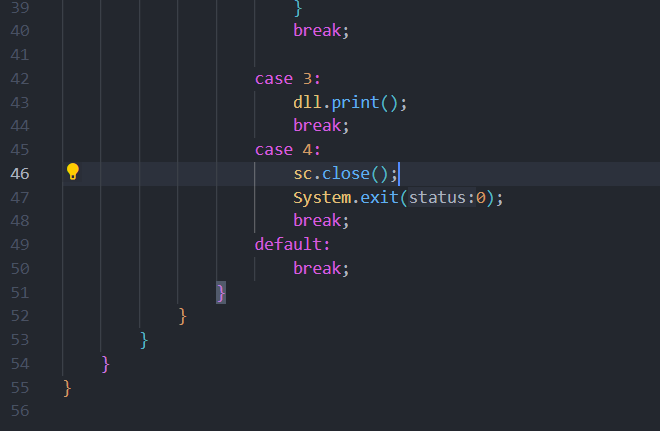




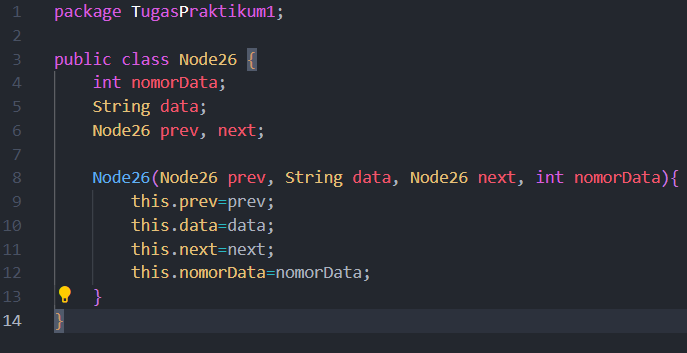


DoubleLinkedListsMain26:





Node26:



* + 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.

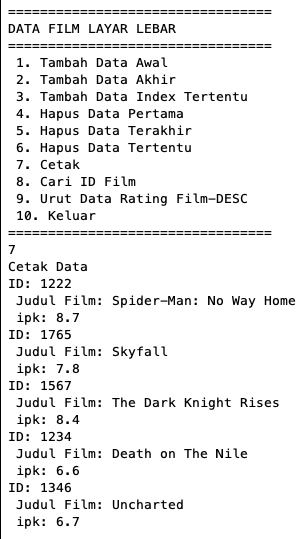
# Contoh Ilustrasi Program

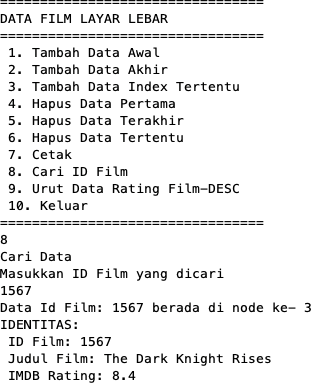
*Menu Awal dan Penambahan Data*

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



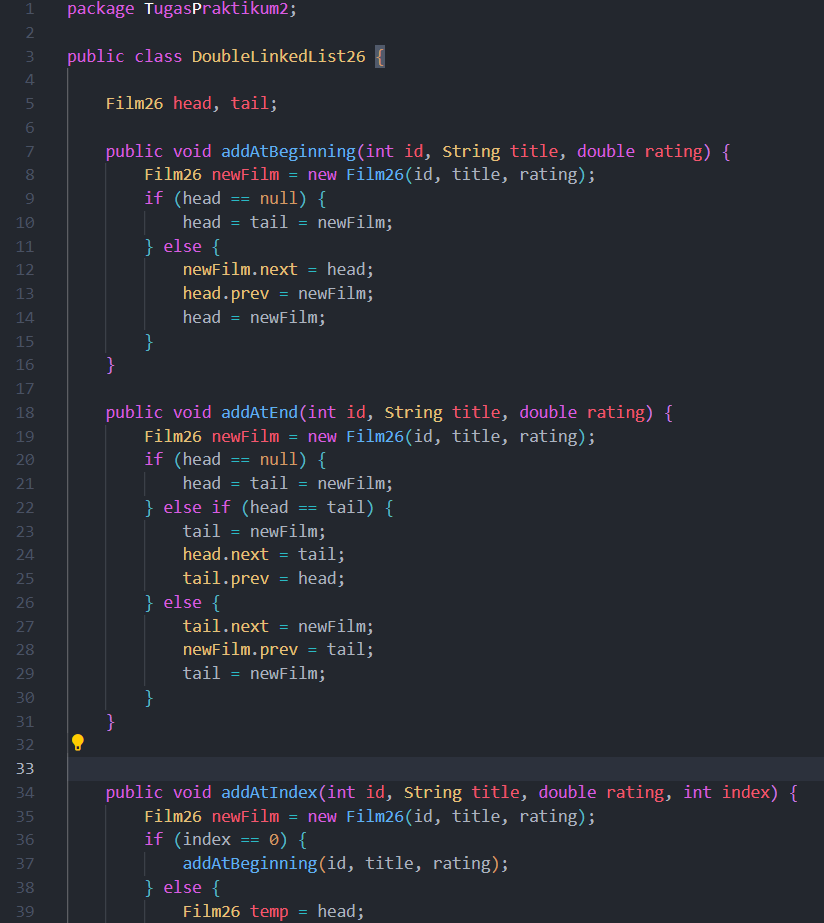
|  |  |
| --- | --- |
|  |  |

*Cetak Data*

*Pencarian Data*

--- \*\*\* ---

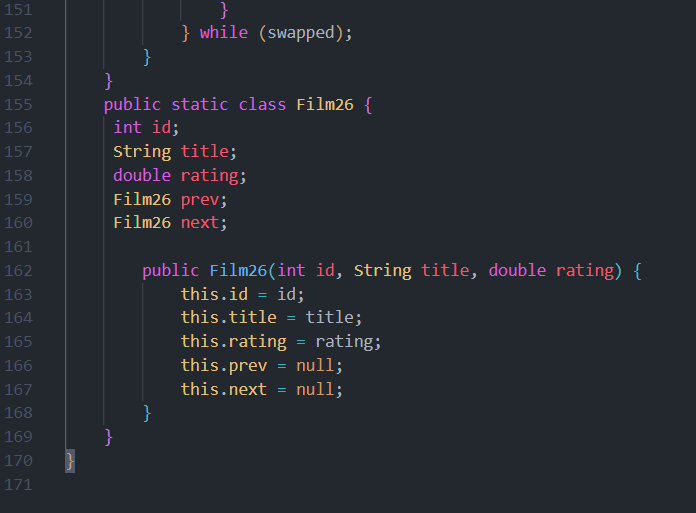
DoubleLinkedList26:











Hasil :

