**LAPORAN PRAKTIKUM JOBSHEET 8**

**DOUBLE LINKED LISTS**

**MATA KULIAH ALGORITMA DAN STRUKTUR DATA**



**Disusun Oleh :**

**Jami’atul Afifah (2341760102)**

**SIB-1F**

**PROGRAM STUDI D4 SISTEM INFOEMASI BISNIS**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2024**

**PERCOBAAN 1**

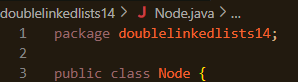
Pada percobaan 1 ini akan dibuat class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan linked list, belakang ataupun indeks tertentu pada linked list).

1. Perhatikan diagram class Node14 dan class DoublelinkedLists14 di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.

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

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

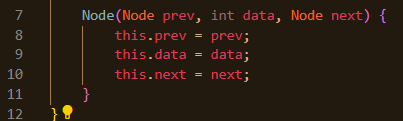
1. Buat paket baru dengan nama doublelinkedlists
2. Buat class di dalam paket tersebut dengan nama Node



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



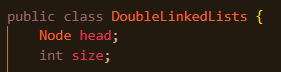
1. Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.



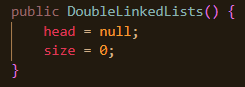
1. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:



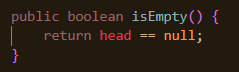
1. Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.



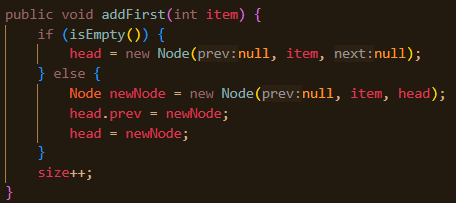
1. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.



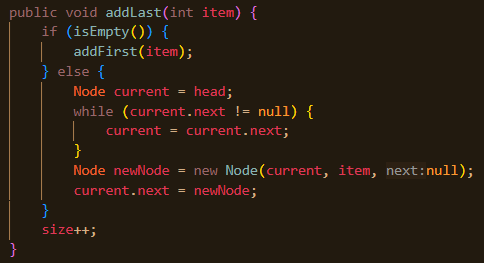
1. Buat method isEmpty(). Method ini digunakan untuk memastikan kondisi linked list kosong.



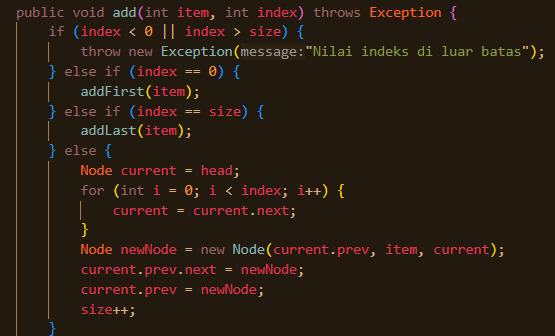
1. Kemudian, buat method addFirst(). Method ini akan menjalankan penambahan data di bagian depan linked list.



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



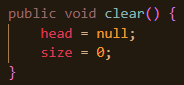
1. Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method add(int item, int index)



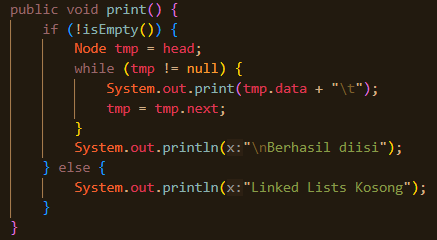
1. Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis, sehingga dapat dibuat method size() untuk mendapatkan nilai dari size.



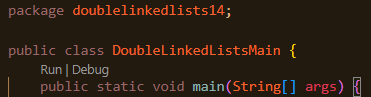
1. Selanjutnya dibuat method clear() untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong.



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.

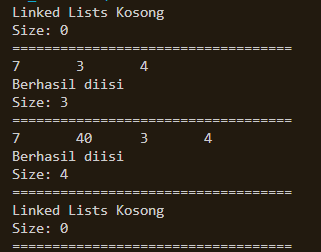


1. Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.



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

**Hasil Output :**



**PERTANYAAN PERCOBAAN**

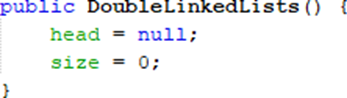
1. Jelaskan perbedaan antara single linked list dengan double linked lists!

* Single Linked List (SLL):
  + Setiap node hanya memiliki satu referensi, yaitu ke node berikutnya (next).
  + Traversal hanya bisa dilakukan ke satu arah, yaitu dari head ke tail.
  + Struktur lebih sederhana dan memerlukan memori lebih sedikit karena hanya menyimpan satu referensi per node.
* Double Linked List (DLL):
  + Setiap node memiliki dua referensi, yaitu ke node sebelumnya (prev) dan ke node berikutnya (next).
  + Traversal bisa dilakukan ke dua arah, yaitu dari head ke tail dan sebaliknya.
  + Struktur lebih kompleks dan memerlukan memori lebih banyak karena menyimpan dua referensi per node.
  + Lebih fleksibel dalam operasi seperti penghapusan dan penambahan node di tengah.

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

* next: Merupakan referensi ke node berikutnya dalam linked list. Ini digunakan untuk traversal maju dari satu node ke node berikutnya.
* prev: Merupakan referensi ke node sebelumnya dalam linked list. Ini digunakan untuk traversal mundur dari satu node ke node sebelumnya.

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



* head = null: Menginisialisasi linked list dalam keadaan kosong dengan tidak ada node awal (head).
* size = 0: Mengatur ukuran linked list menjadi nol karena saat pembuatan linked list baru, belum ada node yang ditambahkan.

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, node tersebut akan menjadi node pertama (head). Karena tidak ada node sebelum node ini, referensi prev diatur menjadi null.

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

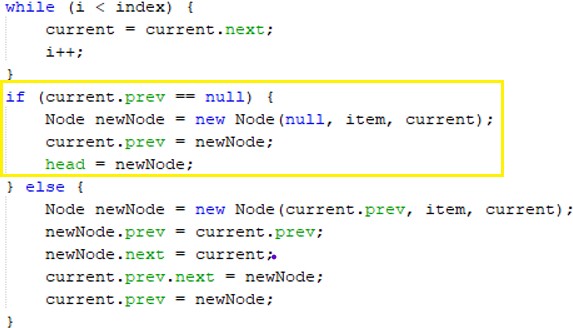
Setelah menambahkan node baru di depan linked list, node baru ini menjadi head yang baru. Statement ini mengatur referensi prev dari node yang sebelumnya menjadi head agar menunjuk ke node baru ini. Hal ini menjaga konsistensi referensi dua arah dalam linked list.

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**);

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

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



jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

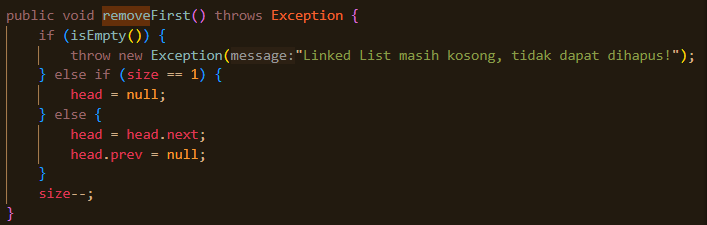
* Loop while (i < index): Melakukan traversal untuk menemukan node pada posisi index.
* if (current.prev == null):
  + Jika current.prev adalah null, berarti current adalah head dari linked list.
  + Membuat node baru (newNode) yang prev-nya adalah null (karena node baru akan menjadi head), data-nya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

**PERCOBAAN 2**

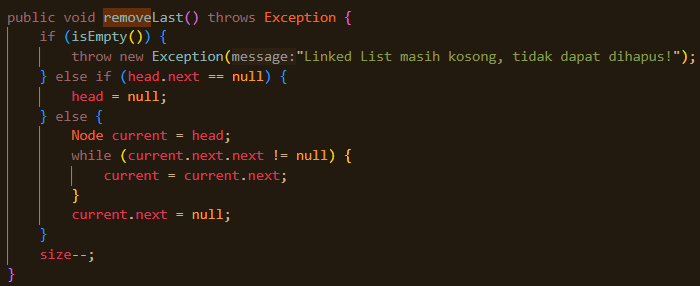
Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists. Method tambahan tersebut akan ditambahkan sesuai pada diagram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

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

1. Buatlah method removeFirst() di dalam class DoubleLinkedLists.



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



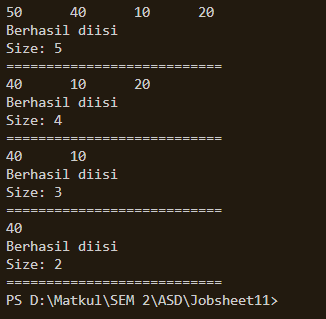
1. Tambahkan pula method remove(int index) pada class DoubleLinkedLists dan amati hasilnya.



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



**Hasil Output:**



**PERTANYAAN PERCOBAAN**

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

head = head.next;

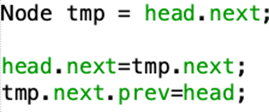
head.prev = null;

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

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

* while (current.next.next != null): Loop ini digunakan untuk traversal melalui linked list sampai menemukan node kedua terakhir. Kondisi current.next.next != null memastikan bahwa current berhenti pada node kedua terakhir.
* Setelah loop selesai, current adalah node kedua terakhir dan current.next adalah node terakhir. Dengan mengatur current.next ke null, kita menghapus referensi ke node terakhir, sehingga node kedua terakhir menjadi node terakhir yang baru.

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



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

Masalah:

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

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



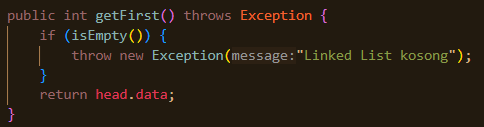
* current.prev.next = current.next;: Statement ini mengatur referensi next dari node sebelumnya (current.prev) untuk melompati current dan menunjuk ke node berikutnya setelah current. Ini secara efektif mengeluarkan current dari linked list dari sisi sebelumnya.
* current.next.prev = current.prev;: Statement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

**PERCOBAAN 3**

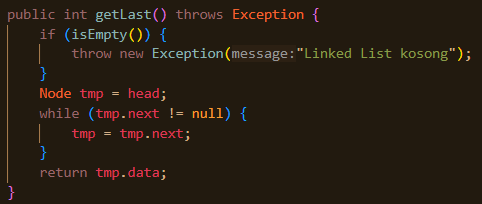
Pada praktikum 3 ini dilakukan uji coba untuk mengambil data pada linked list dalam 3 kondisi, yaitu mengambil data paling awal, paling akhir dan data pada indeks tertentu dalam linked list. Method mengambil data dinamakan dengan get. Ada 3 method get yang dibuat pada praktikum ini sesuai dengan diagram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

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

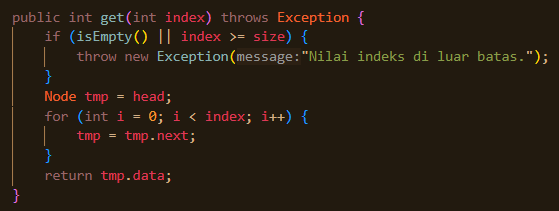
1. Buatlah method getFirst() di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists.



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



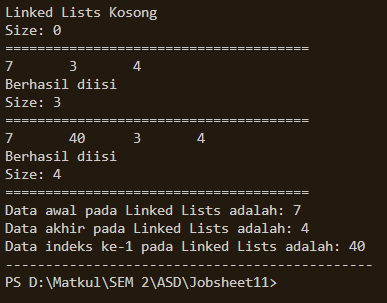
1. Method get(int index) dibuat untuk mendapatkan data pada indeks tertentu



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



**Hasil Output:**



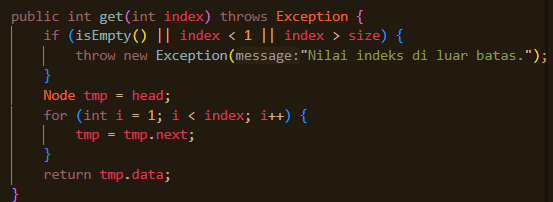
Pertanyaan Percobaan

1. Jelaskan method size() pada class DoubleLinkedLists!

* size adalah atribut dalam class DoubleLinkedLists yang melacak jumlah node dalam linked list.
* Method size() hanya mengembalikan nilai dari atribut size.

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

Biasanya, indeks pada linked list (seperti pada array) dimulai dari 0. Untuk mengatur indeks agar dimulai dari 1, Anda dapat menyesuaikan logika pada operasi terkait indeks. Namun, ini tidak dianjurkan karena akan membuat kode tidak konvensional dan lebih sulit dimengerti. Berikut adalah contoh bagaimana Anda bisa menyesuaikan method `get(int index)` untuk bekerja dengan indeks yang dimulai dari 1:



* Indeks diatur untuk mulai dari 1, sehingga pengecekan batasan indeks (`index < 1` dan `index > size`) disesuaikan.
* Loop dimulai dari 1 dan berjalan hingga `i < index`, bukan `i < index - 1` seperti dalam implementasi dengan indeks yang dimulai dari 0.

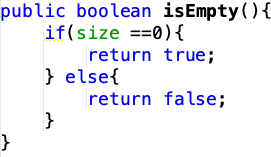
1. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!

**DoubleLinkedLists:**

* + Setiap node memiliki dua referensi: `prev` (mengarah ke node sebelumnya) dan `next` (mengarah ke node berikutnya).
  + Operasi penambahan node (baik di awal, akhir, atau posisi tertentu) melibatkan pengaturan kedua referensi (`prev` dan `next`), sehingga membutuhkan sedikit lebih banyak pekerjaan dibandingkan single linked list.
  + Penghapusan node juga lebih efisien karena dapat langsung mengakses node sebelumnya.

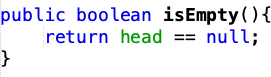
**SingleLinkedLists:**

* + Setiap node hanya memiliki satu referensi: `next` (mengarah ke node berikutnya).
  + Operasi penambahan node (terutama di akhir) mungkin memerlukan traversal dari awal hingga akhir linked list, yang tidak diperlukan dalam double linked list jika ada referensi ke node terakhir.
  + Penghapusan node di posisi tertentu juga membutuhkan traversal karena tidak ada referensi langsung ke node sebelumnya.

1. Jelaskan perbedaan logika dari kedua kode program di bawah ini!
2. 

Metode pertama (a):

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

1. 

Metode kedua (b):

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

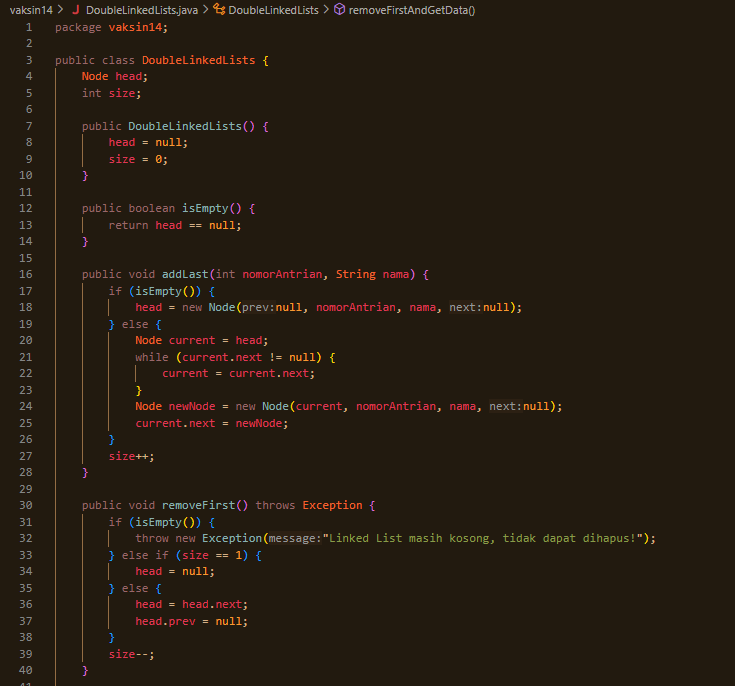
**Perbedaan utama:**

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

**TUGAS PRAKTIKUM**

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

DoubleLinkedList.java



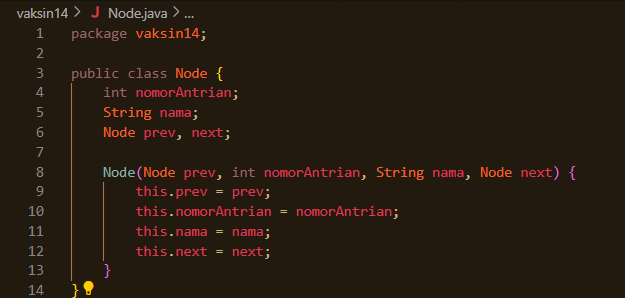


DoubleLinkedListMain.java





Node.java



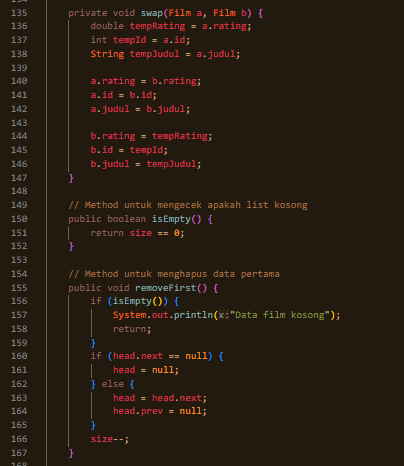
1. Buatlah program daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists, bentuk program memiliki fitur pencarian melalui ID Film dan pengurutan Rating secara descending. Class Film wajib diimplementasikan dalam soal ini.

Film.java

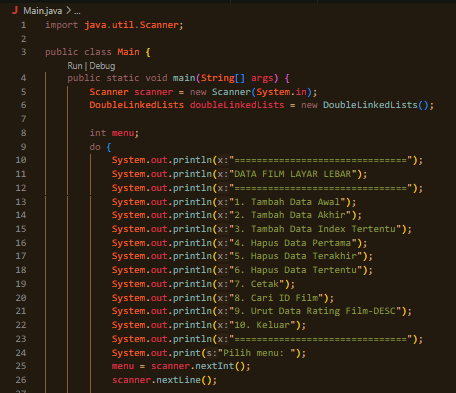


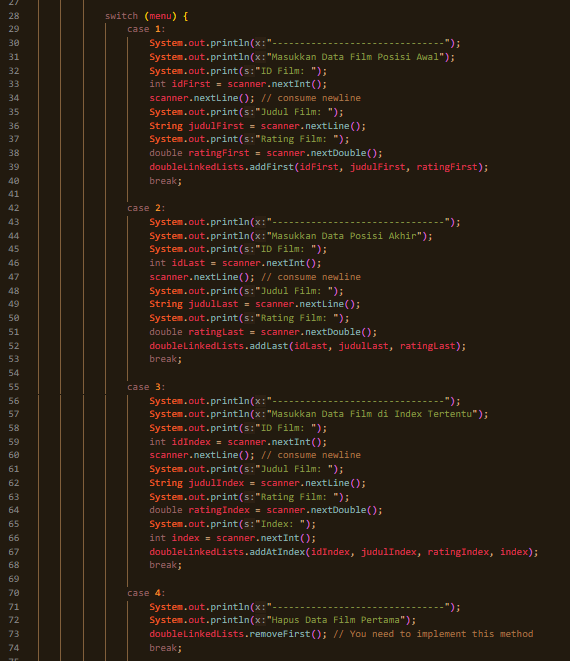






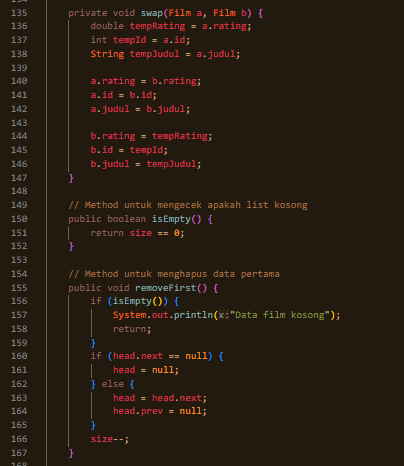
Main.java













Hasil Output:

