LAPORAN PRAKTIKUM STRUKTUR DATA

PEKAN 6



Oleh :

SEPTIAN RIYANDA PUTRA

NIM 2411532016

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI DEPARTEMEN INFORMATIKA UNIVERSITAS ANDALAS

PADANG 2025

1. Pendahuluan

Struktur data merupakan elemen penting dalam pemrograman yang berfungsi untuk menyimpan dan mengelola data secara efisien. Salah satu jenis struktur data yang sering digunakan adalah Linked List. Linked List adalah struktur data linear yang terdiri dari elemen-elemen (disebut simpul atau node) yang saling terhubung melalui pointer atau referensi.

Dalam implementasinya, terdapat beberapa jenis Linked List, salah satunya adalah Double Linked List. Berbeda dengan Single Linked List yang hanya memiliki satu arah (hanya menyimpan referensi ke node berikutnya), Double Linked List menyimpan dua referensi, yaitu ke node sebelumnya (prev) dan ke node berikutnya (next). Hal ini memungkinkan traversal atau penelusuran data dilakukan secara dua arah, baik dari depan ke belakang maupun sebaliknya.

Pada praktikum ini, kita akan mempelajari cara mengimplementasikan dan mengelola Double Linked List menggunakan bahasa pemrograman Java. Praktikum mencakup operasi-operasi dasar seperti penambahan node di awal, tengah, atau akhir, penghapusan node, serta penelusuran (traversal) dan pencarian data dalam list. Dengan memahami konsep dan implementasi Double Linked List, diharapkan kita dapat mengembangkan kemampuan dalam mengelola data dinamis secara efisien dan fleksibel dalam program Java.

1. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. **Memahami konsep dasar Double Linked List**, termasuk struktur node yang memiliki referensi ke node sebelumnya dan node sesudahnya.
2. Mempelajari cara mengimplementasikan *double linked list* menggunakan bahasa pemrograman Java secara manual (tanpa menggunakan library bawaan).
3. Mampu membuat dan memanipulasi node dalam *double linked list*, termasuk operasi tambah, hapus, dan telusur data.
4. Program yang dibuat
5. NodeDLL

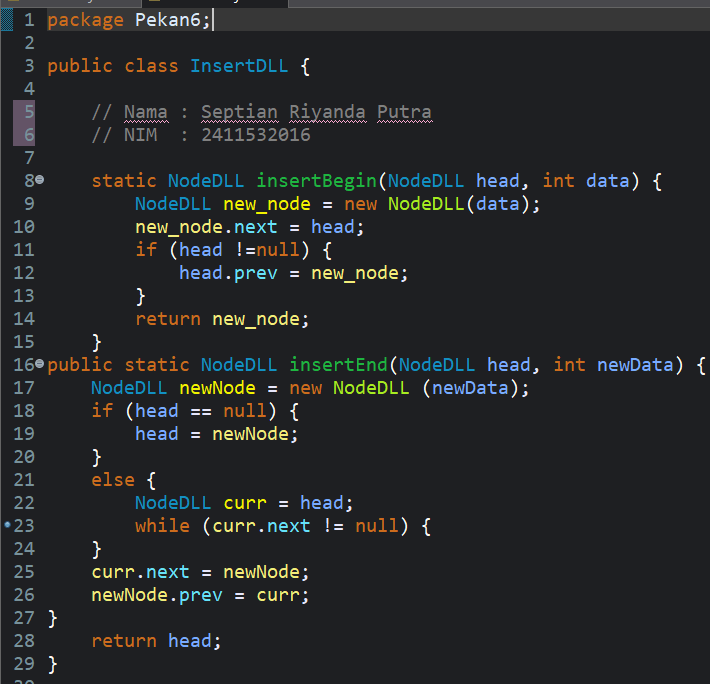


Di dalam kelas ini, terdapat tiga atribut utama:

* int data — menyimpan nilai atau data dari simpul.
* NodeDLL next — menunjuk ke simpul berikutnya.
* NodeDLL prev — menunjuk ke simpul sebelumnya.

Konstruktor NodeDLL(int data) digunakan untuk membuat simpul baru. Ketika sebuah simpul dibuat, nilai data akan diset sesuai parameter, dan referensi next serta prev diinisialisasi dengan null. Hal ini berarti, simpul yang baru dibuat tidak langsung terhubung dengan simpul lain mana pun.

1. InsertDLL



1. **insertBegin(NodeDLL head, int data)**

Metode ini digunakan untuk menambahkan simpul di awal list. Simpul baru akan menjadi simpul pertama (head) dari list, dan simpul sebelumnya (jika ada) akan menunjuk kembali ke simpul baru melalui referensi prev.

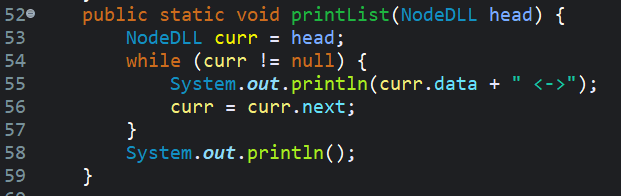
1. **insertEnd(NodeDLL head, int data)**

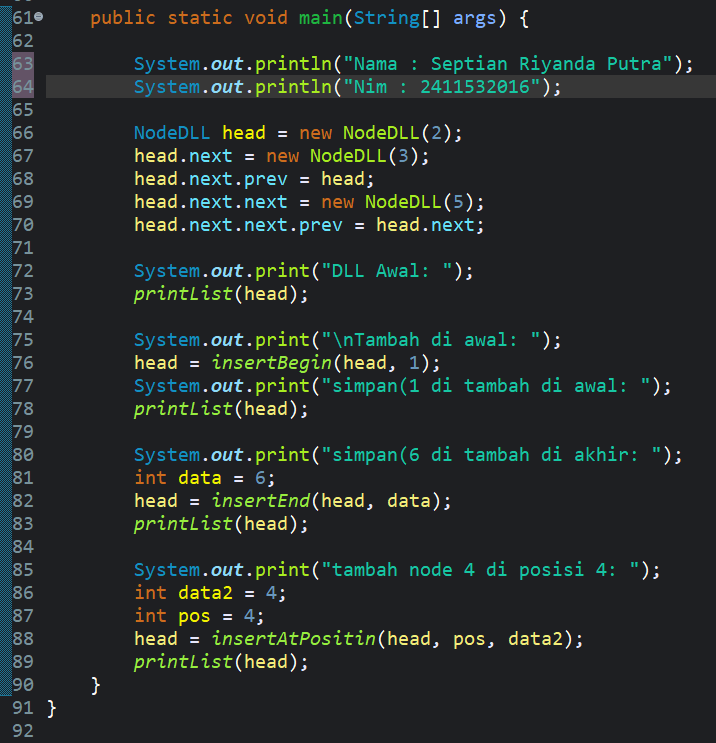
Metode ini digunakan untuk menambahkan simpul di akhir list. Prosesnya melibatkan penelusuran ke simpul terakhir (menggunakan curr.next) dan menyambungkan simpul baru sebagai next dari simpul terakhir tersebut.



1. **insertAtPositin(NodeDLL head, int pos, int new\_data)**

Metode ini digunakan untuk menyisipkan simpul pada posisi tertentu. Posisi dihitung mulai dari 1. Jika posisi yang diberikan adalah 1, maka simpul baru akan ditempatkan di awal. Jika posisi berada di tengah atau akhir, penelusuran akan dilakukan terlebih dahulu untuk menemukan simpul sebelum posisi tersebut.



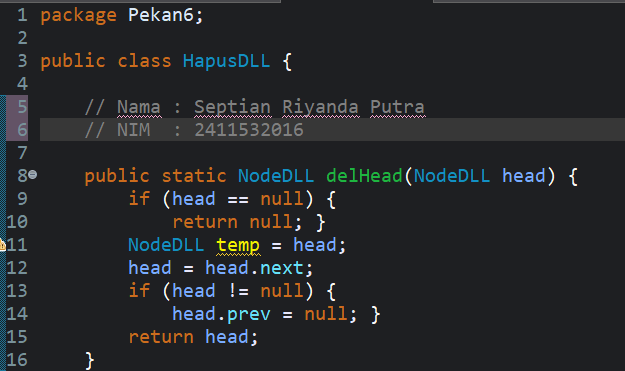


1. **printList(NodeDLL head)**

Metode ini mencetak isi dari Double Linked List dengan cara menelusuri list dari awal hingga akhir menggunakan referensi next.

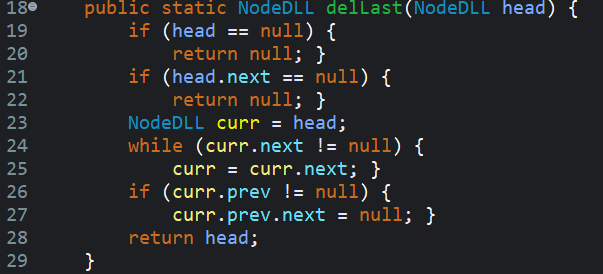
Pada bagian main(), metode-metode di atas diuji dengan membuat list awal berisi 2 → 3 → 5, lalu menyisipkan simpul baru di awal (1), di akhir (6), dan di tengah (4 di posisi ke-4).

1. HapusDLL



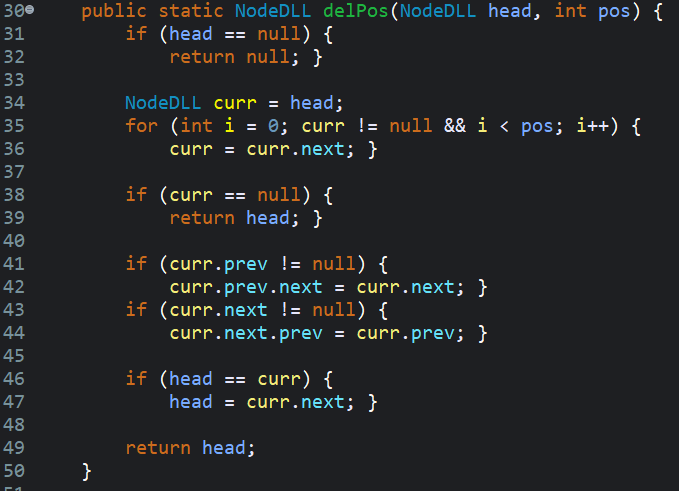
1. **delHead(NodeDLL head)**

Metode ini menghapus simpul pertama dari list. Jika list tidak kosong, maka simpul head akan diubah menjadi simpul setelahnya (head.next) dan referensi prev simpul baru tersebut akan diset ke null.



1. **delLast(NodeDLL head)**

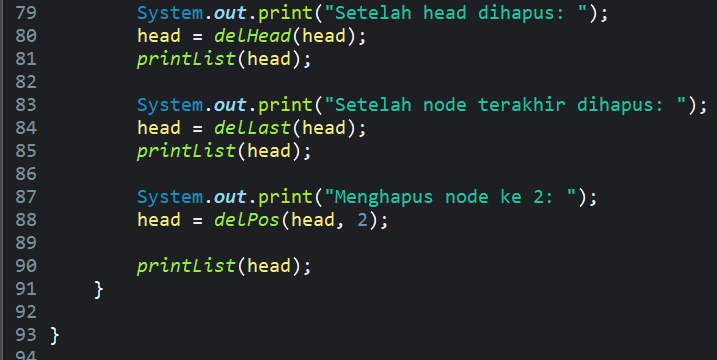
Metode ini menghapus simpul terakhir dari list. Penelusuran dilakukan hingga ke simpul paling akhir (curr.next == null), lalu simpul sebelumnya akan dijadikan simpul terakhir dengan menyetel curr.prev.next ke null.



1. **delPos(NodeDLL head, int pos)**

Metode ini digunakan untuk menghapus simpul pada posisi tertentu. Penelusuran dilakukan sampai ke posisi tersebut. Jika simpul ditemukan, maka simpul sebelumnya dan sesudahnya akan disambungkan langsung, sehingga simpul pada posisi tersebut "terputus" dari list.



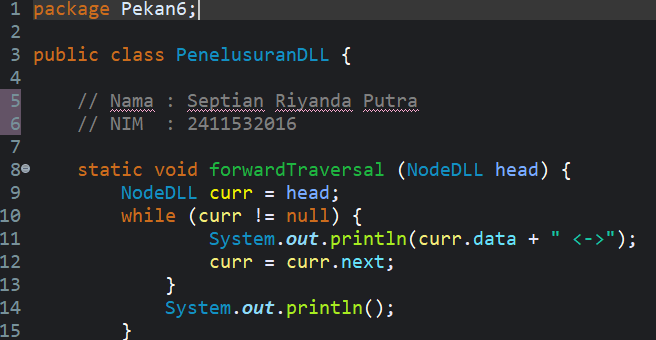


1. **printList(NodeDLL head)**

Sama seperti pada InsertDLL, fungsi ini digunakan untuk mencetak isi dari list. Di dalam main(), list awal terdiri atas 1 → 2 → 3 → 4 → 5, kemudian dilakukan tiga jenis penghapusan:

* Menghapus simpul pertama.
* Menghapus simpul terakhir.
* Menghapus simpul pada posisi ke-2.

1. PenelusuranDLL



1. **forwardTraversal(NodeDLL head)**

Metode ini mencetak semua nilai dalam list dari simpul pertama hingga terakhir dengan menggunakan referensi next



1. backwardTraversal(NodeDLL tail)

Metode ini mencetak semua nilai dari simpul terakhir hingga pertama dengan menggunakan referensi prev.

Di dalam main(), list awal terdiri dari tiga simpul: 1 → 2 → 3. Penelusuran maju dilakukan dari head (simpul pertama), dan penelusuran mundur dilakukan dari tail (simpul terakhir, yaitu simpul ketiga).

1. Kesimpulan

Melalui praktikum ini, dapat disimpulkan bahwa Double Linked List merupakan salah satu struktur data yang sangat fleksibel untuk mengelola data secara dinamis, karena memungkinkan penelusuran dan manipulasi data dari dua arah—maju maupun mundur. Dalam implementasi menggunakan bahasa Java, kita mempelajari bagaimana membangun struktur simpul (node) yang memiliki dua referensi, yaitu ke simpul sebelumnya (prev) dan ke simpul berikutnya (next).

Berbagai operasi dasar pada Double Linked List berhasil diterapkan, antara lain:

1. **Penyisipan data** di awal, di akhir, dan di posisi tertentu dalam list.
2. **Penghapusan data** dari awal, akhir, dan posisi tertentu.
3. **Penelusuran list** secara maju (forward traversal) maupun mundur (backward traversal).

Seluruh operasi tersebut dilakukan tanpa menggunakan library bawaan Java, sehingga memberikan pemahaman mendalam tentang bagaimana data disusun dan dikelola di dalam memori. Praktikum ini juga mengasah kemampuan logika dan analisis dalam mengatur hubungan antar node agar tidak terjadi kesalahan referensi seperti null pointer atau broken link.

Dengan memahami dan mengimplementasikan Double Linked List secara manual, kita memperoleh dasar yang kuat untuk mempelajari struktur data yang lebih kompleks serta meningkatkan keterampilan dalam pemrograman berbasis objek.