LAPORAN PRAKTIKUM

STRUKTUR DATA



DISUSUN OLEH :

Sasya Zamora

2411533014

DOSEN PENGAMPU :

Dr. Wahyudi, S.T.M.T

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

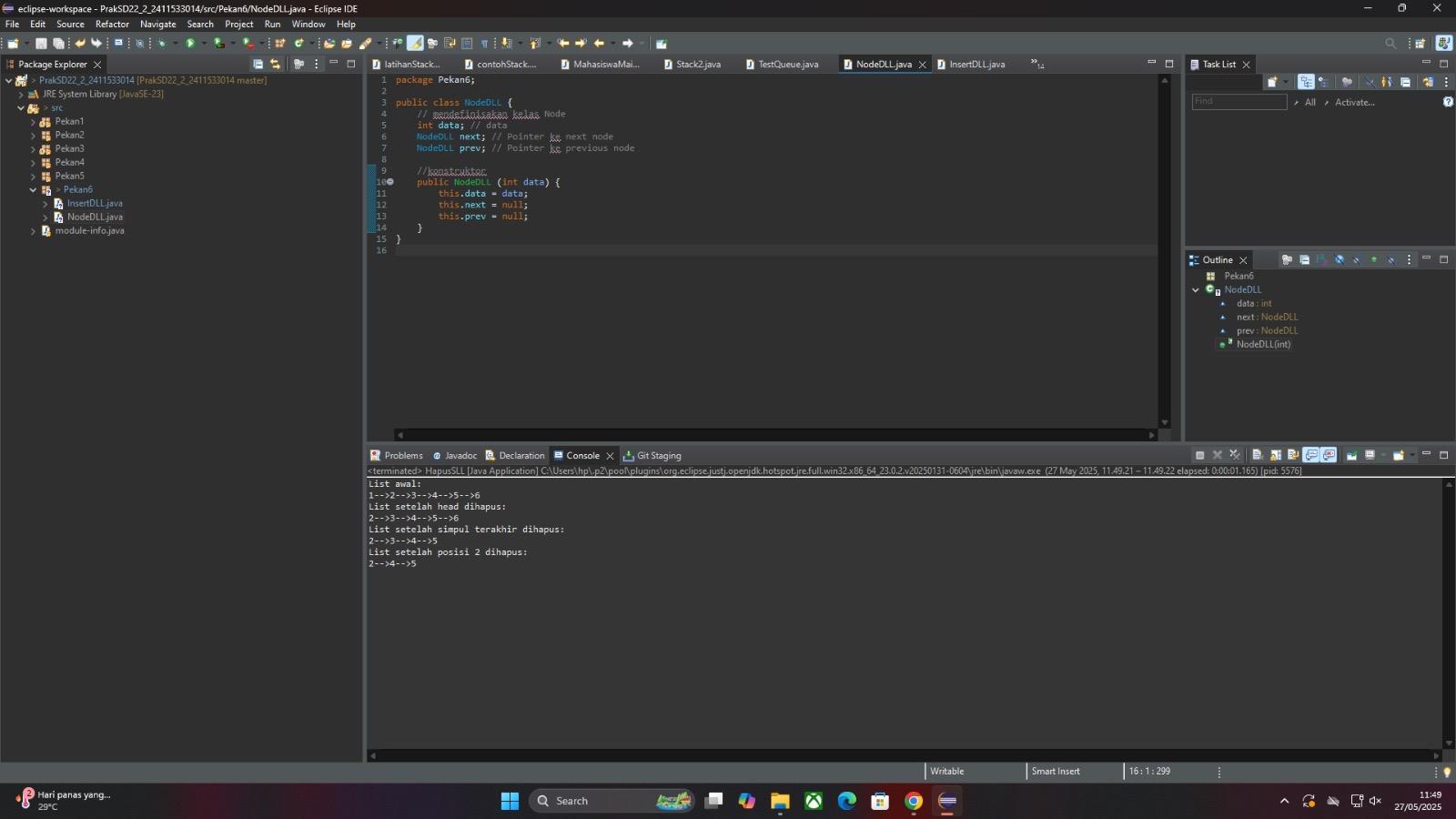
2025

1. TUJUAN
2. Mahasiswa mampu menguasai Implementasi Doubly Linked List
3. Mahasiswa mampu mempelajari Operasi Penyisipan dan Penghapusan
4. Mahasiswa mampu memahami Traversal Doubly Linked List
5. PEMBAHASAN

Pratikum ini memfokuskan pada implementasi dan penggunaan struktur data doubly linked list di Java. Mahasiswa diharuskan untuk membangun fungsi-fungsi dasar yang meliputi pembuatan node, penyisipan node di awal, akhir, dan pada posisi tertentu dalam daftar, serta penghapusan node dari awal, akhir, dan posisi tertentu. Proses ini melibatkan pemahaman yang mendalam tentang mekanisme pointer next dan prev yang menghubungkan setiap node, serta bagaimana manipulasi pointer tersebut memengaruhi struktur keseluruhan doubly linked list. Selain itu, pratikum juga mencakup implementasi fungsi untuk melakukan traversal (penelusuran) daftar baik secara maju maupun mundur, memperkuat pemahaman tentang akses dan pengolahan data dalam struktur data ini.

Melalui implementasi dan pengujian fungsi-fungsi tersebut, mahasiswa mempraktikkan konsep-konsep kunci dalam pemrograman berorientasi objek dan pengelolaan memori. Mereka belajar bagaimana membangun struktur data yang kompleks dari komponen-komponen yang lebih sederhana, dan bagaimana merancang fungsi yang efisien dan mudah dipahami. Pratikum ini memberikan pengalaman praktis yang berharga dalam penerapan teori struktur data ke dalam kode program yang fungsional, memperkuat pemahaman konseptual melalui penerapan langsung.

1. LANGKAH PRAKTIKUM
2. Buka Eclipse IDE
3. Lalu, buat package pekan6
4. Kemudian, buat class dan kodingan seperti dibawah ini :

* NodeDLL

Langkah 1: Deklarasi Package

package Pekan6;

Baris ini mendeklarasikan bahwa kelas NodeDLL termasuk dalam paket bernama Pekan6. Paket digunakan untuk mengorganisir kelas dan mencegah konflik penamaan.

Langkah 2: Definisi Kelas

public class NodeDLL {

Baris ini mendefinisikan kelas publik bernama NodeDLL. Kata kunci public berarti kelas ini dapat diakses dari kelas lain mana pun. NodeDLL kemungkinan kependekan dari "Node Doubly Linked List," yang menunjukkan bahwa kelas ini mewakili satu node dalam struktur data doubly linked list.

Langkah 3: Anggota Data

int data; // data

NodeDLL next; // Penunjuk ke node berikutnya

NodeDLL prev; // Penunjuk ke node sebelumnya

Baris-baris ini mendeklarasikan variabel instans (anggota data) dari kelas NodeDLL:

• int data;: Variabel integer untuk menyimpan data yang terkait dengan node ini.

• NodeDLL next;: Referensi ke node berikutnya dalam daftar tertaut. Tipe datanya adalah NodeDLL karena menunjuk ke objek NodeDLL lainnya.

• NodeDLL prev;: Referensi ke node sebelumnya dalam daftar tertaut. Inilah yang membuatnya menjadi doubly linked list.

Langkah 4: Konstruktor

//konstruktor

public NodeDLL(int data) {

this.data = data;

this.next = null;

this.prev = null;

}

Ini adalah konstruktor untuk kelas NodeDLL. Ia menerima integer data sebagai input dan menginisialisasi node:

• this.data = data;: Menugaskan data input ke anggota data node. this merujuk pada objek saat ini yang sedang dibuat.

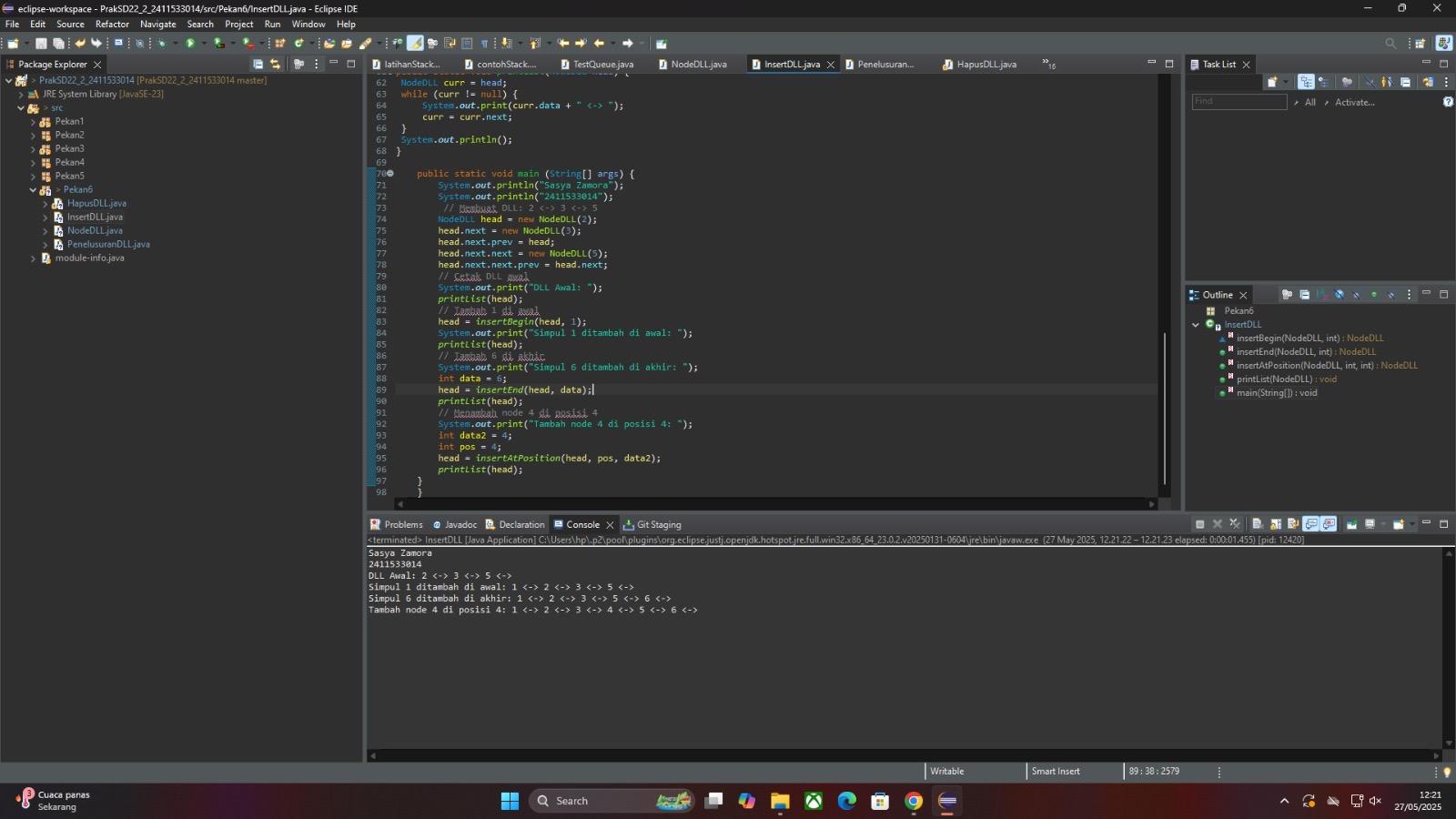
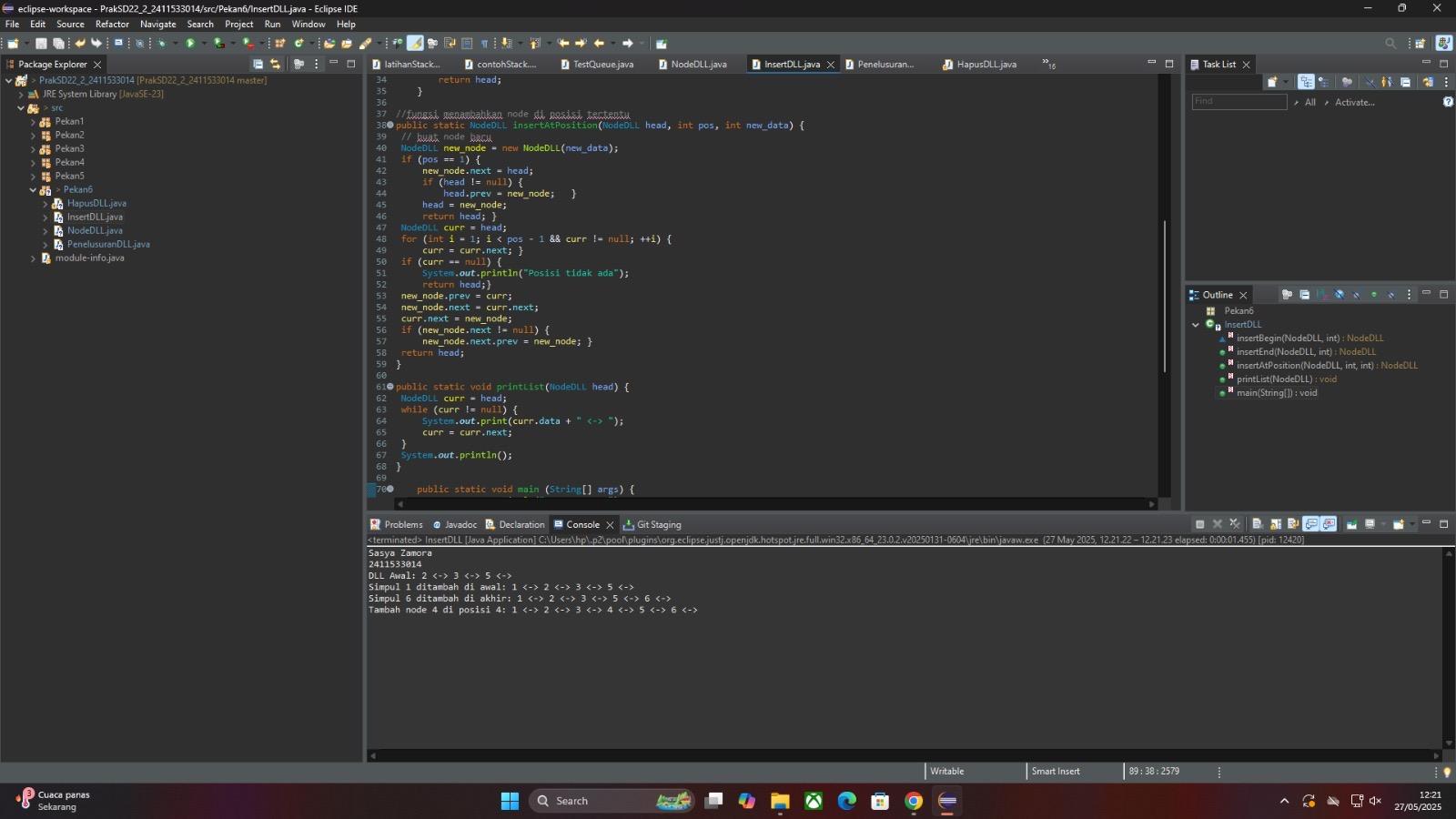
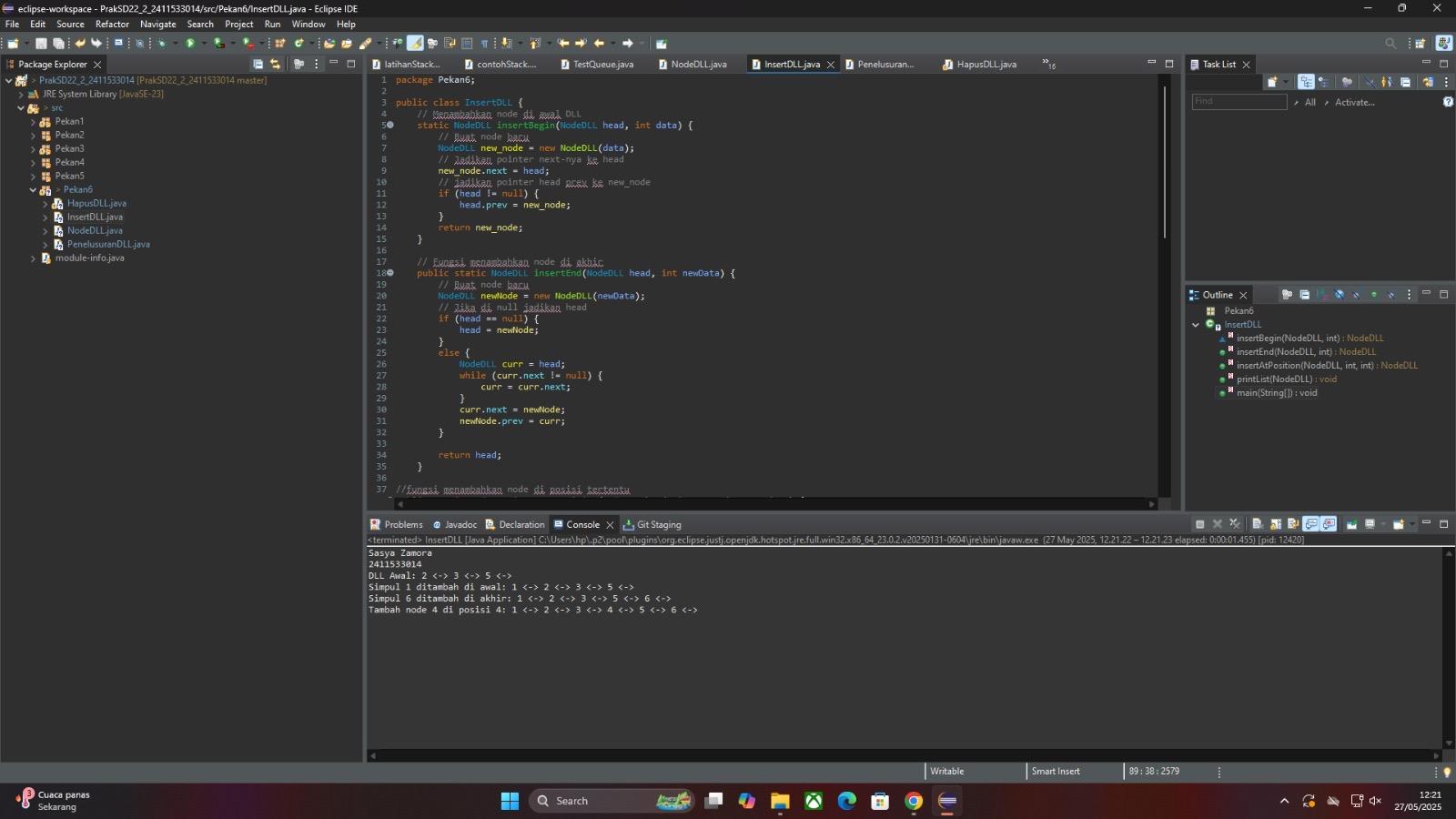
• this.next = null;: Mengatur penunjuk next ke null, yang menunjukkan bahwa node ini belum terhubung ke node berikutnya.

• this.prev = null;: Mengatur penunjuk prev ke null, yang menunjukkan bahwa node ini belum terhubung ke node sebelumnya.

Langkah 5: Kurung Kurawal Penutup

}

Kurung kurawal penutup ini menandai akhir dari definisi kelas NodeDLL.

* InsertDLL

1. NodeDLL.java (Kelas Node):

File ini kemungkinan besar mendefinisikan kelas NodeDLL, yang merupakan representasi dari sebuah node dalam doubly linked list. Setiap node akan memiliki tiga atribut utama:

• data: Menyimpan data integer dari node tersebut.

• next: Pointer (penunjuk) ke node berikutnya dalam daftar. Bernilai null jika node tersebut adalah node terakhir.

• prev: Pointer ke node sebelumnya dalam daftar. Bernilai null jika node tersebut adalah node pertama.

2. InsertDLL.java (Kelas InsertDLL):

File ini berisi implementasi fungsi-fungsi untuk manipulasi doubly linked list:

• insertBegin(NodeDLL head, int data): Fungsi ini menambahkan sebuah node baru di awal daftar.

◦ Parameter: head (NodeDLL): Menunjuk ke node pertama (kepala) daftar. data (int): Data integer yang akan disimpan dalam node baru.

◦ Cara Kerja:

1. Membuat node baru dengan data yang diberikan.

2. Menghubungkan node baru ke node yang sebelumnya merupakan kepala daftar (new\_node.next = head).

3. Jika daftar tidak kosong, memperbarui pointer prev dari node kepala sebelumnya untuk menunjuk ke node baru (head.prev = new\_node).

4. Memperbarui head agar menunjuk ke node baru.

5. Mengembalikan head yang sudah diperbarui.

• insertEnd(NodeDLL head, int newData): Fungsi ini menambahkan sebuah node baru di akhir daftar.

◦ Parameter: head (NodeDLL): Menunjuk ke node pertama (kepala) daftar. newData (int): Data integer yang akan disimpan dalam node baru.

◦ Cara Kerja:

1. Membuat node baru dengan data yang diberikan.

2. Jika daftar kosong (head == null), node baru menjadi kepala daftar.

3. Jika daftar tidak kosong, iterasi melalui daftar sampai menemukan node terakhir (node dengan curr.next == null).

4. Menghubungkan node baru ke node terakhir (curr.next = newNode) dan menghubungkan node baru ke node sebelumnya (newNode.prev = curr).

5. Mengembalikan head.

• insertAtPosition(NodeDLL head, int position, int data): (Tidak terlihat di gambar, tetapi kemungkinan ada). Fungsi ini akan menambahkan node baru pada posisi tertentu dalam daftar. Ini akan memerlukan iterasi hingga posisi yang ditentukan sebelum melakukan penyisipan.

• printList(NodeDLL head): (Tidak terlihat di gambar, tetapi kemungkinan ada). Fungsi ini akan mencetak isi dari seluruh doubly linked list.

• main(String[] args): Fungsi utama program. Fungsi ini akan digunakan untuk menguji fungsi-fungsi lain yang telah dibuat. Output di konsol menunjukkan contoh penggunaan insertBegin dan insertEnd.

3. insertAtPosition(NodeDLL head, int pos, int newData): Fungsi ini menambahkan node baru pada posisi tertentu dalam doubly linked list.

• Parameter:

◦ head: NodeDLL, menunjuk ke node pertama (kepala) dari daftar.

◦ pos: int, posisi di mana node baru akan disisipkan (posisi 1 adalah kepala daftar).

◦ newData: int, data yang akan disimpan dalam node baru.

• Cara Kerja:

1. Membuat Node Baru: Sebuah node baru (newNode) dibuat dengan data newData.

2. Menangani Kasus Daftar Kosong: Jika daftar kosong (head == null), dan posisi yang diminta adalah 1, maka node baru menjadi kepala daftar.

3. Mencari Posisi: Kode menggunakan sebuah perulangan for untuk menelusuri daftar sampai mencapai posisi (pos) yang diinginkan. curr digunakan sebagai penunjuk untuk iterasi. Perhatikan kondisi (curr != null && i < pos -1):

◦ curr != null: Memastikan bahwa kita tidak mencoba mengakses node setelah akhir daftar.

◦ i < pos - 1: Menghitung posisi. Kita berhenti satu node sebelum posisi yang dituju karena kita akan menyisipkan node sebelum node pada posisi pos.

4. Menangani Posisi Tidak Valid: Jika perulangan selesai dan curr masih null (artinya posisi tidak ada dalam daftar), pesan kesalahan dicetak, dan fungsi mengembalikan head tanpa perubahan.

5. Menyisipkan Node: Node baru disisipkan dengan menghubungkan pointer next dan prev dengan benar:

◦ newNode.next = curr.next;

◦ newNode.prev = curr;

◦ curr.next.prev = newNode;

◦ curr.next = newNode;

6. Mengembalikan Head: Fungsi mengembalikan head yang telah diperbarui.

4. printList(NodeDLL head): Fungsi ini mencetak isi dari doubly linked list.

• Parameter: head: NodeDLL, menunjuk ke node pertama (kepala) daftar.

• Cara Kerja: Ia melakukan iterasi melalui daftar menggunakan curr, mencetak data dari setiap node sampai mencapai akhir daftar (curr == null).

5. main(String[] args): Fungsi utama untuk pengujian. Contoh di konsol menunjukkan penggunaan berbagai fungsi, termasuk insertAtPosition.

Fungsi main(String[] args):

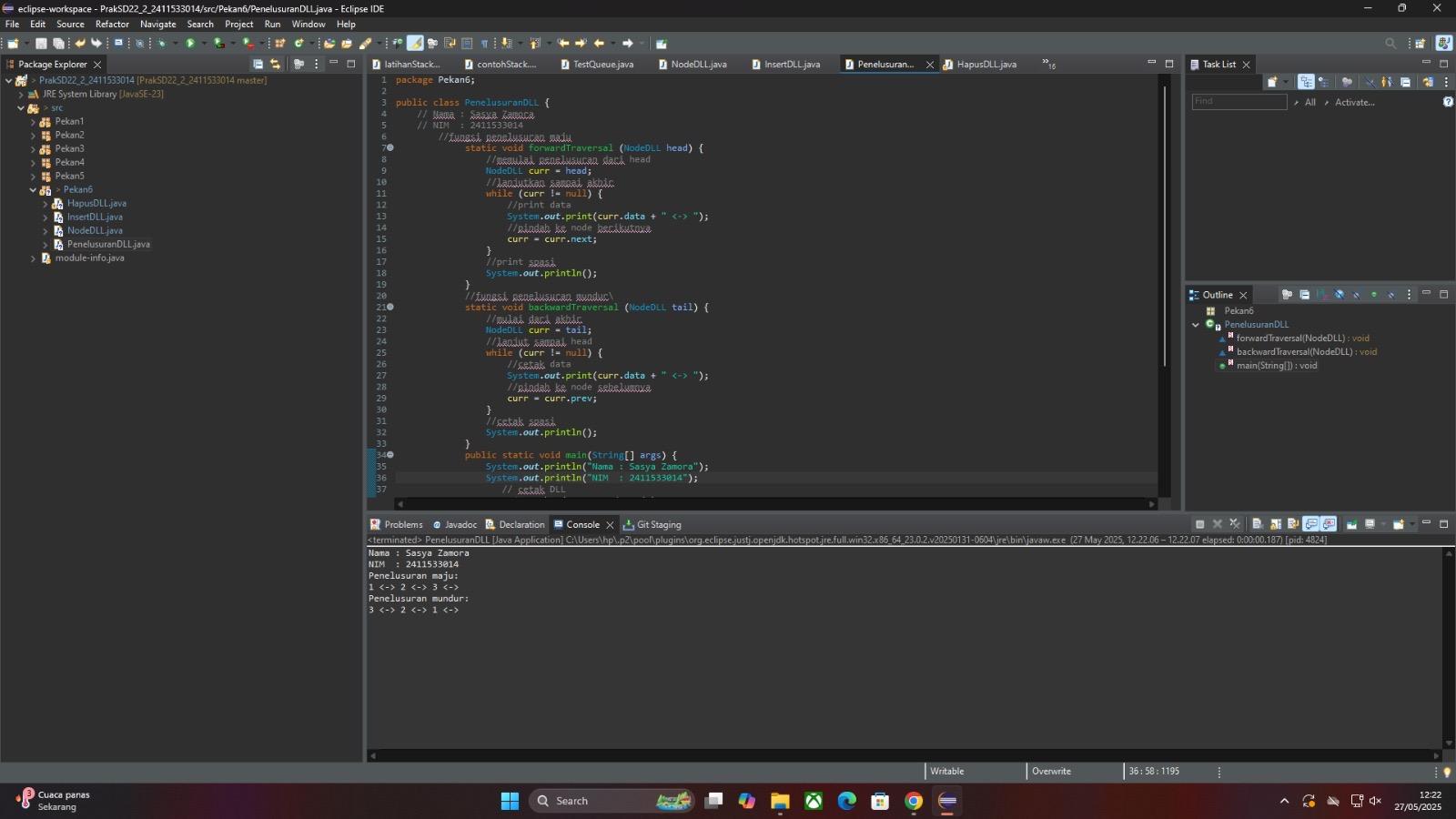
Fungsi ini bertindak sebagai titik masuk utama program. Ia melakukan hal-hal berikut:

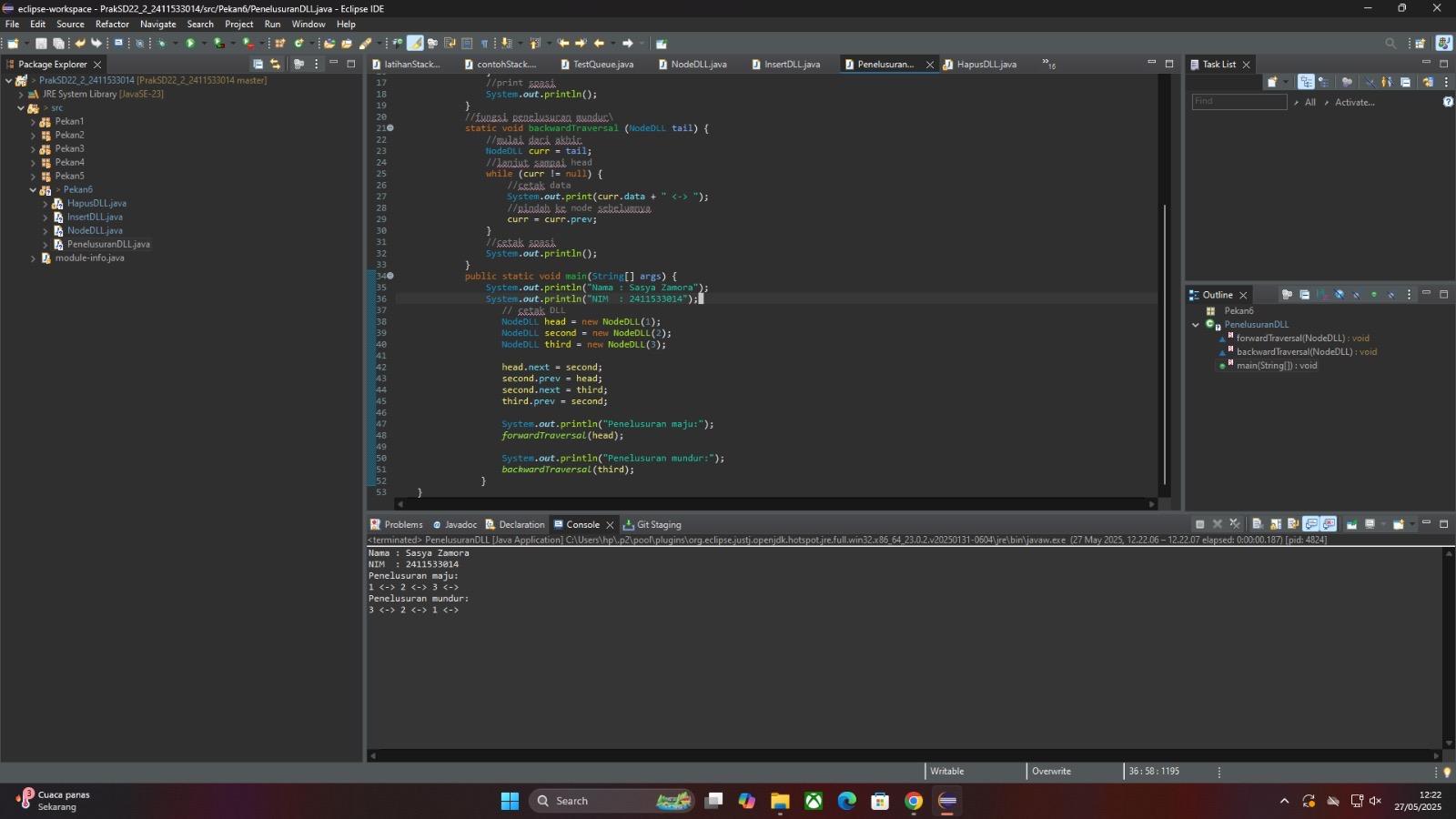
1. Inisialisasi: Membuat doubly linked list awal dengan beberapa node (2, 3, 5). Perhatikan bagaimana node dihubungkan secara manual menggunakan pointer next dan prev.

2. Menambahkan Node di Awal: Memanggil insertBegin untuk menambahkan node dengan data 1 di awal daftar. Kemudian, daftar dicetak untuk menunjukkan hasilnya.

3. Menambahkan Node di Akhir: Memanggil insertEnd untuk menambahkan node dengan data 6 di akhir daftar. Daftar dicetak lagi.

4. Menambahkan Node di Posisi Tertentu: Memanggil insertAtPosition untuk menambahkan node dengan data 4 pada posisi 4. Ini adalah pengujian untuk fungsi yang baru ditambahkan. Daftar dicetak.

* PenelusuranDLL



1. forwardTraversal(NodeDLL head): Fungsi ini melakukan traversal maju (dari kepala ke ekor) pada doubly linked list.

• Parameter: head: NodeDLL, menunjuk ke node pertama (kepala) dari daftar.

• Cara Kerja: Fungsi menggunakan perulangan while untuk iterasi melalui daftar. curr dimulai dari head. Selama curr bukan null (belum mencapai akhir daftar), data dari node saat ini dicetak, dan curr diperbarui ke node berikutnya (curr.next).

2. backwardTraversal(NodeDLL tail): Fungsi ini melakukan traversal mundur (dari ekor ke kepala) pada doubly linked list.

• Parameter: tail: NodeDLL, menunjuk ke node terakhir (ekor) dari daftar.

• Cara Kerja: Mirip dengan forwardTraversal, tetapi iterasi dilakukan dari belakang. curr dimulai dari tail. Selama curr bukan null, data dari node saat ini dicetak, dan curr diperbarui ke node sebelumnya (curr.prev).

3. main(String[] args): Fungsi utama yang menunjukkan penggunaan kedua fungsi traversal.

• Cara Kerja:

◦ Daftar diinisialisasi dengan beberapa node (manual).

◦ forwardTraversal dipanggil untuk mencetak daftar dari depan ke belakang.

◦ backwardTraversal dipanggil untuk mencetak daftar dari belakang ke depan. Perhatikan bahwa tail harus dicari terlebih dahulu sebelum memanggil fungsi ini (menggunakan iterasi ke node terakhir).

Fungsi main(String[] args):

Fungsi main ini melakukan hal berikut:

1. Inisialisasi Doubly Linked List: Ia membuat sebuah doubly linked list sederhana dengan tiga node (berisi data 1, 2, dan 3). Perhatikan bagaimana pointer next dan prev diatur secara manual untuk menghubungkan node-node tersebut:

NodeDLL head = new NodeDLL(1);

NodeDLL second = new NodeDLL(2);

NodeDLL third = new NodeDLL(3);

head.next = second;

second.prev = head;

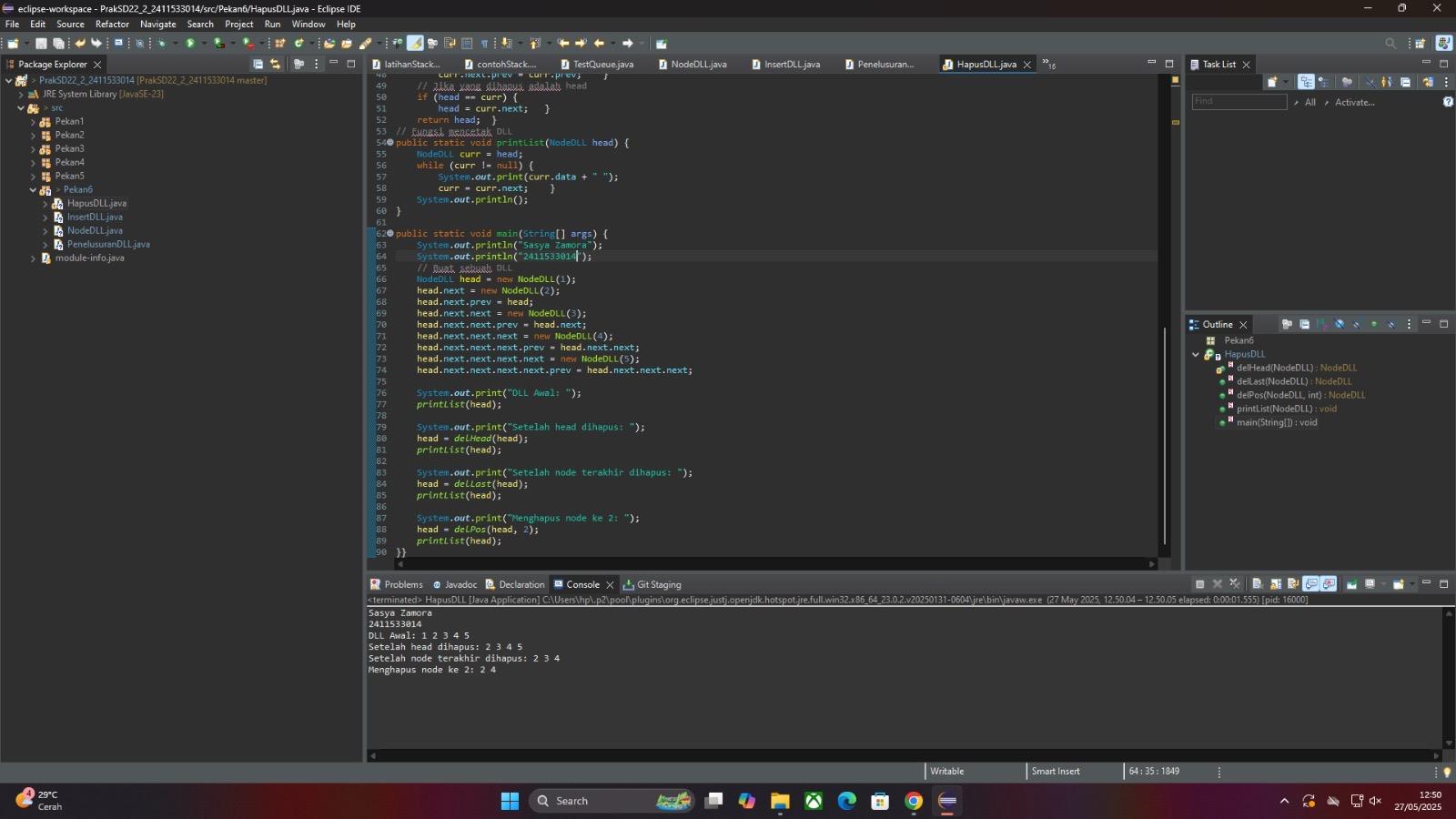
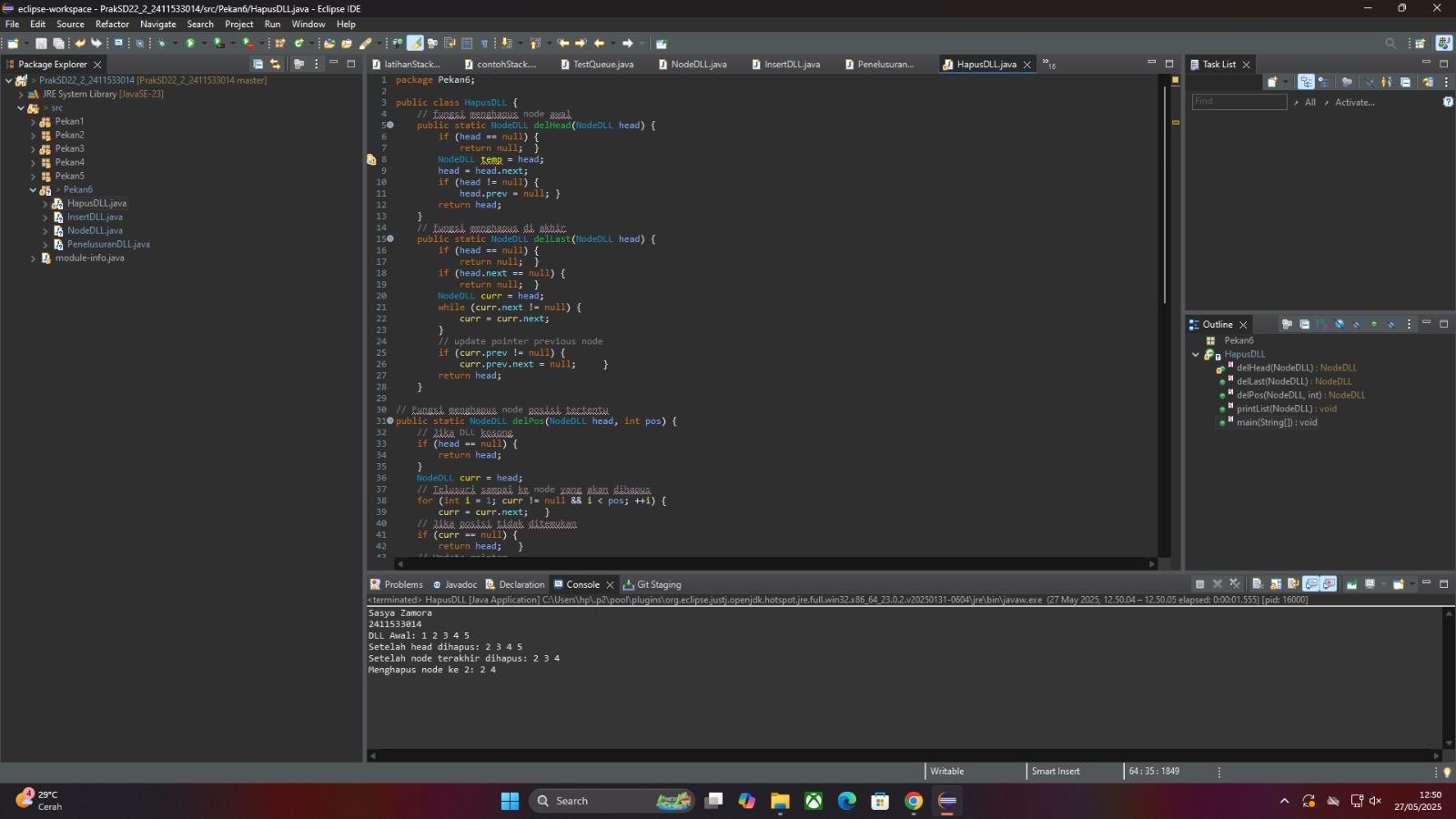
second.next = third;

third.prev = second;

2. Traversal Maju: Memanggil fungsi forwardTraversal dengan head sebagai argumen. Fungsi ini akan mencetak data node-node dari awal hingga akhir daftar.

3. Traversal Mundur: Memanggil fungsi backwardTraversal dengan third (node terakhir) sebagai argumen. Fungsi ini akan mencetak data node-node dari akhir hingga awal daftar.

* HapusDLL



1. delHead(NodeDLL head): Fungsi ini menghapus node pertama (kepala) dari doubly linked list.

• Parameter: head: NodeDLL, menunjuk ke node pertama (kepala) daftar.

• Cara Kerja:

◦ Jika daftar kosong (head == null), fungsi mengembalikan null.

◦ Jika daftar hanya memiliki satu node (head.next == null), head diset ke null, dan node tersebut dihapus.

◦ Jika daftar memiliki lebih dari satu node, pointer prev dari node kedua diset ke null, dan head diperbarui ke node kedua.

◦ Fungsi mengembalikan head yang telah diperbarui.

2. delLast(NodeDLL head): Fungsi ini menghapus node terakhir dari doubly linked list.

• Parameter: head: NodeDLL, menunjuk ke node pertama (kepala) daftar.

• Cara Kerja:

◦ Jika daftar kosong (head == null), fungsi mengembalikan null.

◦ Jika daftar hanya memiliki satu node (head.next == null), head diset ke null, dan node tersebut dihapus.

◦ Jika daftar memiliki lebih dari satu node, fungsi melakukan iterasi hingga node terakhir (curr.next == null). Kemudian, pointer next dari node sebelumnya diset ke null, dan pointer prev dari node terakhir diset ke null.

◦ Fungsi mengembalikan head.

3. delPos(NodeDLL head, int pos): Fungsi ini menghapus node pada posisi tertentu dari doubly linked list.

• Parameter: head: NodeDLL, menunjuk ke node pertama (kepala) daftar. pos: int, posisi node yang akan dihapus (posisi 1 adalah kepala daftar).

• Cara Kerja:

◦ Jika daftar kosong (head == null) atau posisi tidak valid (pos <= 0), fungsi mengembalikan head tanpa perubahan.

◦ Fungsi melakukan iterasi hingga node pada posisi pos. Kemudian, pointer next dan prev dari node sekitarnya diperbarui untuk melewatkan node yang akan dihapus.

◦ Fungsi mengembalikan head.

4. printList(NodeDLL head): Fungsi ini mencetak isi dari doubly linked list. (Fungsi ini sama seperti di kode sebelumnya).

5. main(String[] args): Fungsi utama yang menunjukkan penggunaan fungsi-fungsi penghapusan.

• Cara Kerja:

◦ Daftar diinisialisasi dengan beberapa node.

◦ delHead dipanggil untuk menghapus node pertama.

◦ delLast dipanggil untuk menghapus node terakhir.

◦ delPos dipanggil untuk menghapus node pada posisi tertentu.

◦ Daftar dicetak setelah setiap operasi penghapusan untuk menunjukkan hasilnya.

Fungsi main(String[] args):

Fungsi main ini melakukan hal berikut:

1. Inisialisasi Doubly Linked List: Ia membuat sebuah doubly linked list dengan empat node (1, 2, 3, 4). Node-node dihubungkan secara manual menggunakan pointer next dan prev.

2. Mencetak Daftar Awal: printList(head) dipanggil untuk mencetak isi daftar awal.

3. Menghapus Node Pertama: delHead(head) dipanggil untuk menghapus node pertama (node dengan data 1). Hasilnya dicetak.

4. Menghapus Node Terakhir: delLast(head) dipanggil untuk menghapus node terakhir (node dengan data yang awalnya 4, sekarang 3). Hasilnya dicetak.

5. Menghapus Node di Posisi Tertentu: delPos(head, 2) dipanggil untuk menghapus node pada posisi kedua (node dengan data yang awalnya 2, sekarang 3). Hasilnya dicetak.

1. KESIMPULAN

Pratikum implementasi doubly linked list ini berhasil memberikan pemahaman yang komprehensif tentang struktur data tersebut, meliputi implementasi, manipulasi, dan traversal. Mahasiswa mampu mengimplementasikan fungsi penyisipan dan penghapusan node, serta traversal maju dan mundur. Proses debugging dan perbaikan kode meningkatkan kemampuan analisis dan pemecahan masalah, menekankan pentingnya penanganan kesalahan dan validasi input untuk menghasilkan kode yang robust dan andal. Secara keseluruhan, pratikum ini memberikan pengalaman praktis yang berharga dalam memahami dan mengaplikasikan konsep dasar struktur data dalam pemrograman.