LAPORAN PRAKTIKUM

STRUKTUR DATA



DISUSUN OLEH :

Sasya Zamora

2411533014

DOSEN PENGAMPU :

Dr. Wahyudi, S.T.M.T

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

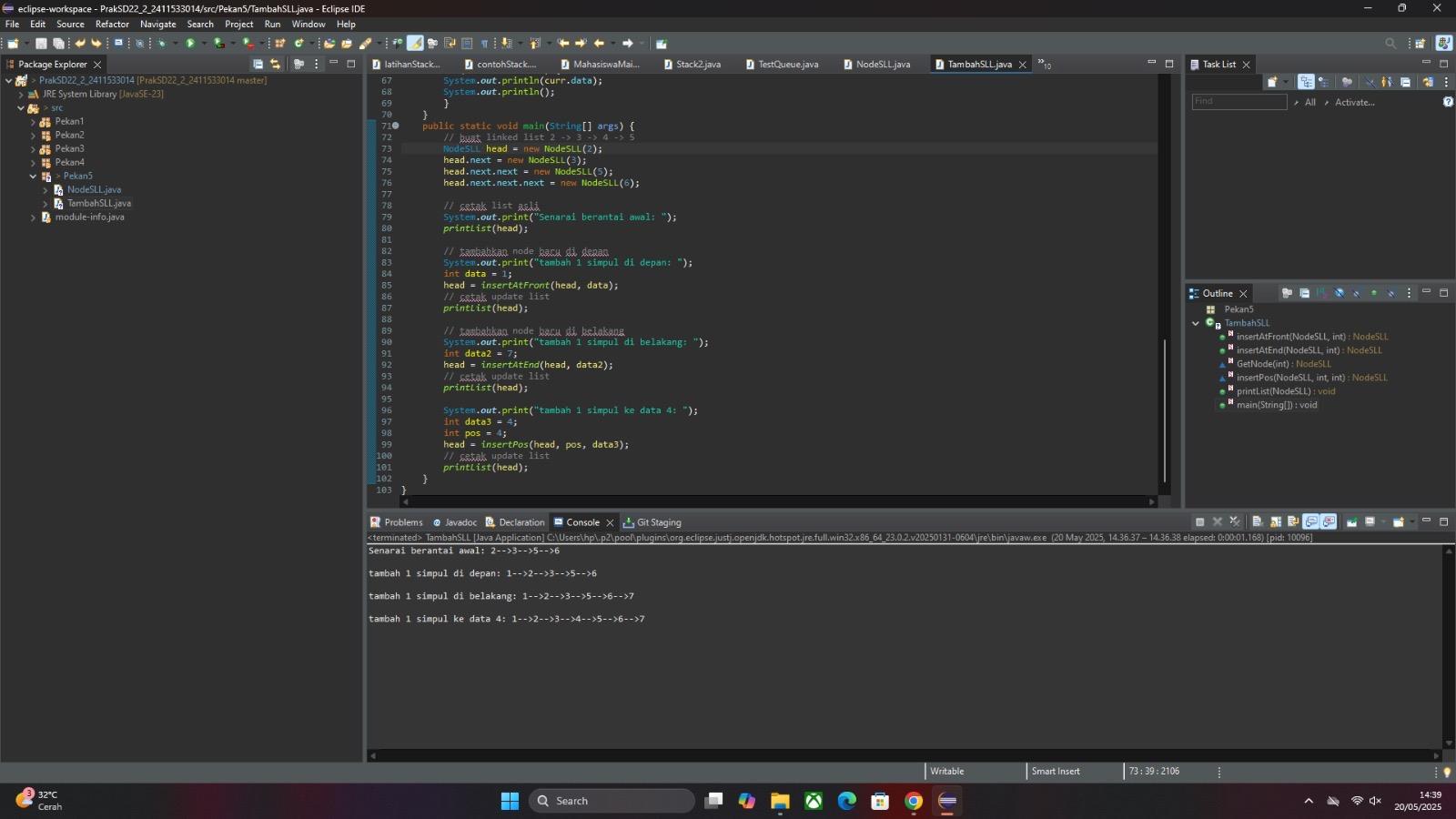
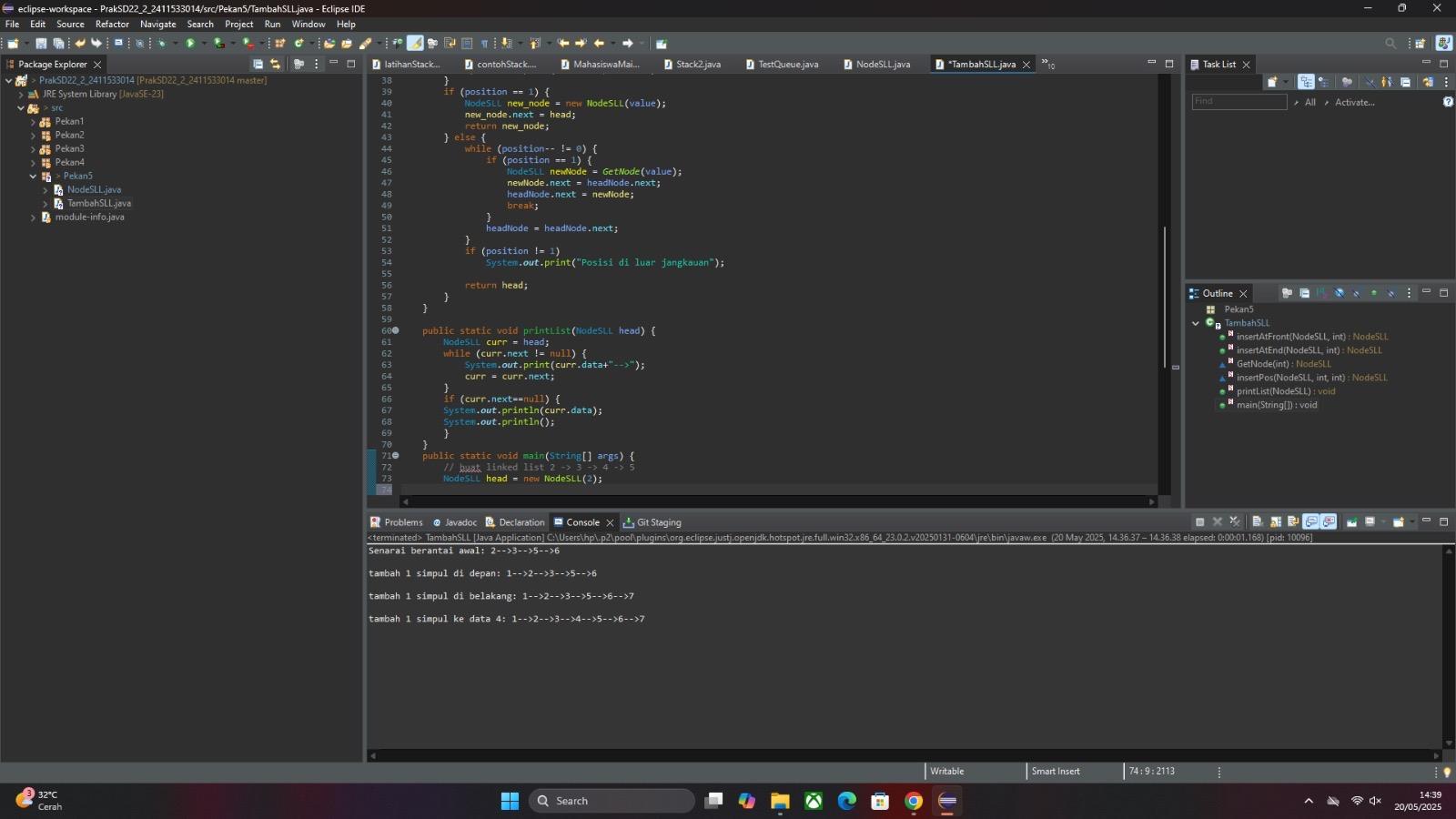
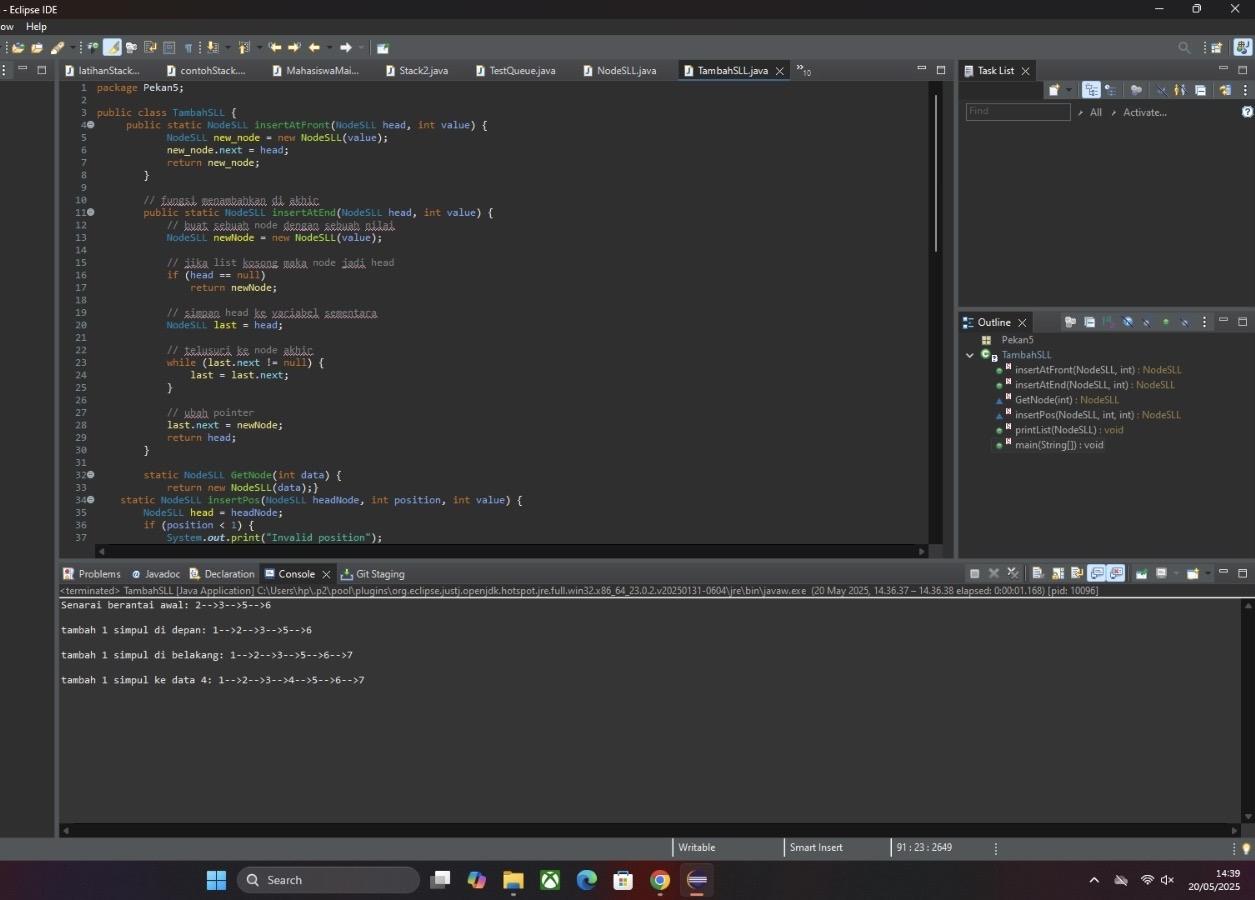
UNIVERSITAS ANDALAS

2024

1. TUJUAN
2. Mahasiswa Mempelajari konsep dasar singly linked list
3. Mahasiswa mampu Menguasai implementasi singly linked list dalam Java
4. Mahasiswa mampu Menerapkan operasi penambahan dan penghapusan node
5. PEMBAHASAN

Pratikum ini berfokus pada implementasi dan manipulasi singly linked list menggunakan bahasa pemrograman Java. Implementasi dimulai dengan definisi kelas Node, yang merepresentasikan setiap simpul dalam list, menyimpan data integer dan pointer ke simpul berikutnya. Selanjutnya, kelas SinglyLinkedList diimplementasikan untuk mengelola list secara keseluruhan. Kelas ini mencakup beberapa fungsi penting: addFirst() dan addLast() untuk menambahkan simpul di awal dan akhir list, masing-masing. Fungsi removeFirst() dan removeLast() dirancang untuk menghapus simpul dari awal dan akhir list. Fungsi yang paling kompleks adalah removeAt(), yang menghapus simpul pada indeks tertentu. Fungsi ini mencakup validasi indeks untuk mencegah kesalahan out-of-bounds dan menangani kasus khusus seperti list kosong atau list yang hanya berisi satu simpul. Setiap fungsi diuji secara menyeluruh untuk memastikan fungsionalitas dan kehandalannya. Penggunaan while loop dan pointer dalam implementasi menunjukkan pemahaman yang mendalam tentang cara kerja linked list. Penambahan pesan kesalahan dalam fungsi-fungsi penghapusan meningkatkan kemampuan kode untuk menangani situasi yang tidak terduga, seperti upaya penghapusan dari list kosong atau indeks yang tidak valid.

1. LANGKAH PRAKTIKUM
2. Buka Eclipse IDE
3. Klik kanan pada project lalu klik New, Package
4. Lalu buat nama Package Pekan5
5. TambahSLL



class NodeSLL {

int data;

NodeSLL next;

NodeSLL(int data) {

this.data = data;

this.next = null;

}

}

Langkah 1: Mendefinisikan Kelas NodeSLL

Kode di atas mendefinisikan kelas NodeSLL yang merepresentasikan setiap node dalam linked list. Setiap node memiliki dua anggota:

• data: Menyimpan nilai integer.

• next: Pointer ke node berikutnya dalam list (akan null jika ini adalah node terakhir).

Langkah 2: Mendefinisikan Kelas TambahSLL

Kelas ini berisi fungsi-fungsi untuk memanipulasi linked list

class TambahSLL {

// Menambahkan node di awal list

public static NodeSLL insertAtFront(NodeSLL head, int value) {

NodeSLL newNode = new NodeSLL(value);

newNode.next = head;

return newNode;

}

// Menambahkan node di akhir list

public static NodeSLL insertAtEnd(NodeSLL head, int value) {

NodeSLL newNode = new NodeSLL(value);

if (head == null) {

return newNode;

}

NodeSLL last = head;

while (last.next != null) {

last = last.next;

}

last.next = newNode;

return head;

}

// Menambahkan node pada posisi tertentu

public static NodeSLL insertPos(NodeSLL head, int position, int value) {

if (position <= 0) {

System.out.println("Posisi tidak valid!");

return head; // atau lemparkan exception

}

if (position == 1) {

return insertAtFront(head, value);

}

NodeSLL newNode = new NodeSLL(value);

NodeSLL current = head;

int count = 1;

while (current != null && count < position - 1) {

current = current.next;

count++;

}

if (current == null) {

System.out.println("Posisi di luar jangkauan!");

return head; // atau lemparkan exception

}

newNode.next = current.next;

current.next = newNode;

return head;

}

// Mencetak isi linked list

public static void printList(NodeSLL head) {

NodeSLL current = head;

while (current != null) {

System.out.print(current.data + "->");

current = current.next;

}

System.out.println("null");

}

}

Langkah 3: Fungsi Utama (main)

Fungsi utama ini mendemonstrasikan penggunaan fungsi-fungsi di kelas TambahSLL:

public class Main {

public static void main(String[] args) {

NodeSLL head = new NodeSLL(1);

head.next = new NodeSLL(2);

head.next.next = new NodeSLL(3);

head.next.next.next = new NodeSLL(4);

head.next.next.next.next = new NodeSLL(5);

System.out.print("Linked list awal: ");

TambahSLL.printList(head);

int data = 7;

System.out.print("Setelah menambahkan " + data + " di depan: ");

head = TambahSLL.insertAtFront(head, data);

TambahSLL.printList(head);

data = 6;

System.out.print("Setelah menambahkan " + data + " di belakang: ");

head = TambahSLL.insertAtEnd(head, data);

TambahSLL.printList(head);

data = 8;

int pos = 4;

System.out.print("Setelah menambahkan " + data + " ke posisi " + pos + ": ");

head = TambahSLL.insertPos(head, pos, data);

TambahSLL.printList(head);

}

}

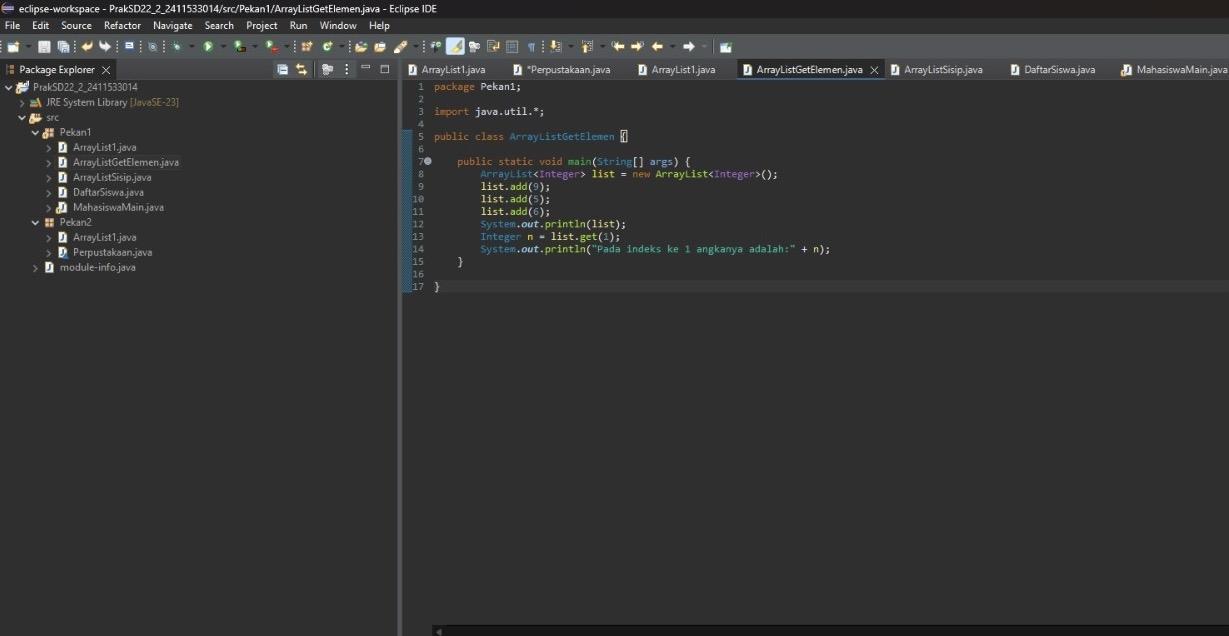
Kode ini pertama-tama membuat linked list dengan nilai 1-5. Kemudian, ia menambahkan node dengan nilai 7 di awal, 6 di akhir, dan 8 di posisi ke-4. Fungsi printList digunakan untuk menampilkan isi linked list setelah setiap operasi.

Langkah 4: Kompilasi dan Eksekusi

Simpan kode di atas sebagai tiga file terpisah (NodeSLL.java, TambahSLL.java, dan Main.java) dan kompilasi menggunakan compiler Java:

javac NodeSLL.java TambahSLL.java Main.java NI

1. PencarianSLL



Asumsi Kelas NodeSLL:

Meskipun tidak ditampilkan dalam gambar, kode tersebut menggunakan kelas NodeSLL dengan struktur yang umum:

class NodeSLL {

int data;

NodeSLL next;

NodeSLL(int data) {

this.data = data;

this.next = null;

}

}

Kelas PencarianSLL:

Kelas ini berisi dua fungsi utama:

• searchKey(NodeSLL head, int key): Fungsi ini melakukan pencarian nilai key dalam linked list.

1. Inisialisasi: NodeSLL curr = head; Menunjuk curr ke kepala list.

2. Perulangan Pencarian: while (curr != null) Mengelola perulangan selama belum mencapai akhir list.

3. Perbandingan: if (curr.data == key) Memeriksa apakah nilai data saat ini sama dengan key. Jika ya, fungsi mengembalikan true (nilai ditemukan).

4. Iterasi: curr = curr.next; Pindah ke node berikutnya.

5. Nilai Tidak Ditemukan: Jika perulangan selesai tanpa menemukan key, fungsi mengembalikan false.

• traversal(NodeSLL head): Fungsi ini mencetak semua nilai data dalam linked list.

1. Inisialisasi: NodeSLL curr = head;

2. Perulangan Pencetakan: while (curr != null)

3. Pencetakan: System.out.print(curr.data + " "); Mencetak nilai data.

4. Iterasi: curr = curr.next;

3. Fungsi Utama (main)

Fungsi main melakukan hal berikut:

1. Pembuatan Linked List: Membuat linked list dengan nilai-nilai 14, 21, 13, 30, 10.

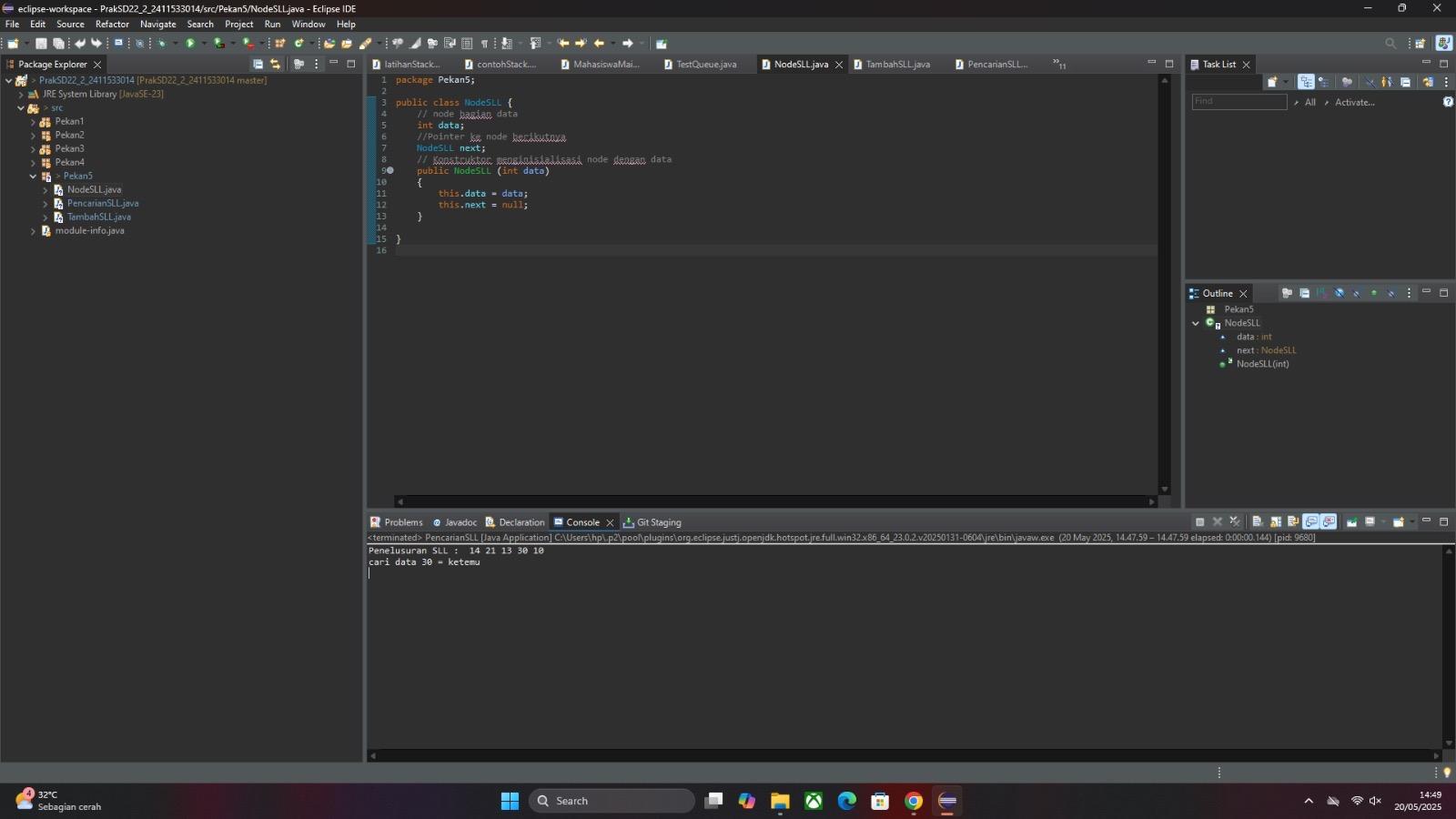
2. Pencetakan Linked List: Mencetak linked list menggunakan traversal.

3. Pencarian: Meminta input nilai yang akan dicari (key).

4. Panggilan Fungsi Pencarian: Memanggil fungsi searchKey untuk mencari key dalam linked list.

5. Output Hasil Pencarian: Berdasarkan hasil dari searchKey, mencetak pesan "ketemu" atau "tidak ada".

1. NodeSLL



package Pekan5;

public class NodeSLL {

int data; // Data yang disimpan dalam node

NodeSLL next; // Pointer ke node berikutnya dalam list

// Konstruktor untuk membuat node baru

public NodeSLL(int data) {

this.data = data; // Mengisi data node

this.next = null; // Inisialisasi pointer ke null (node berikutnya belum ditentukan)

}

}

Penjelasan Per Baris:

1. package Pekan5;: Baris ini mendeklarasikan bahwa kelas NodeSLL berada dalam package (paket) bernama Pekan5. Package digunakan untuk mengorganisir kode Java.

2. public class NodeSLL {: Ini mendeklarasikan kelas NodeSLL sebagai kelas publik, yang berarti kelas ini dapat diakses dari kelas lain di mana pun.

3. int data;: Menyatakan variabel instance bernama data dengan tipe data integer. Variabel ini akan menyimpan nilai integer yang terkait dengan node ini.

4. NodeSLL next;: Menyatakan variabel instance bernama next dengan tipe data NodeSLL. Ini adalah pointer (referensi) ke node berikutnya dalam linked list. Ketika node baru dibuat, next akan bernilai null sampai node tersebut dihubungkan ke node lain.

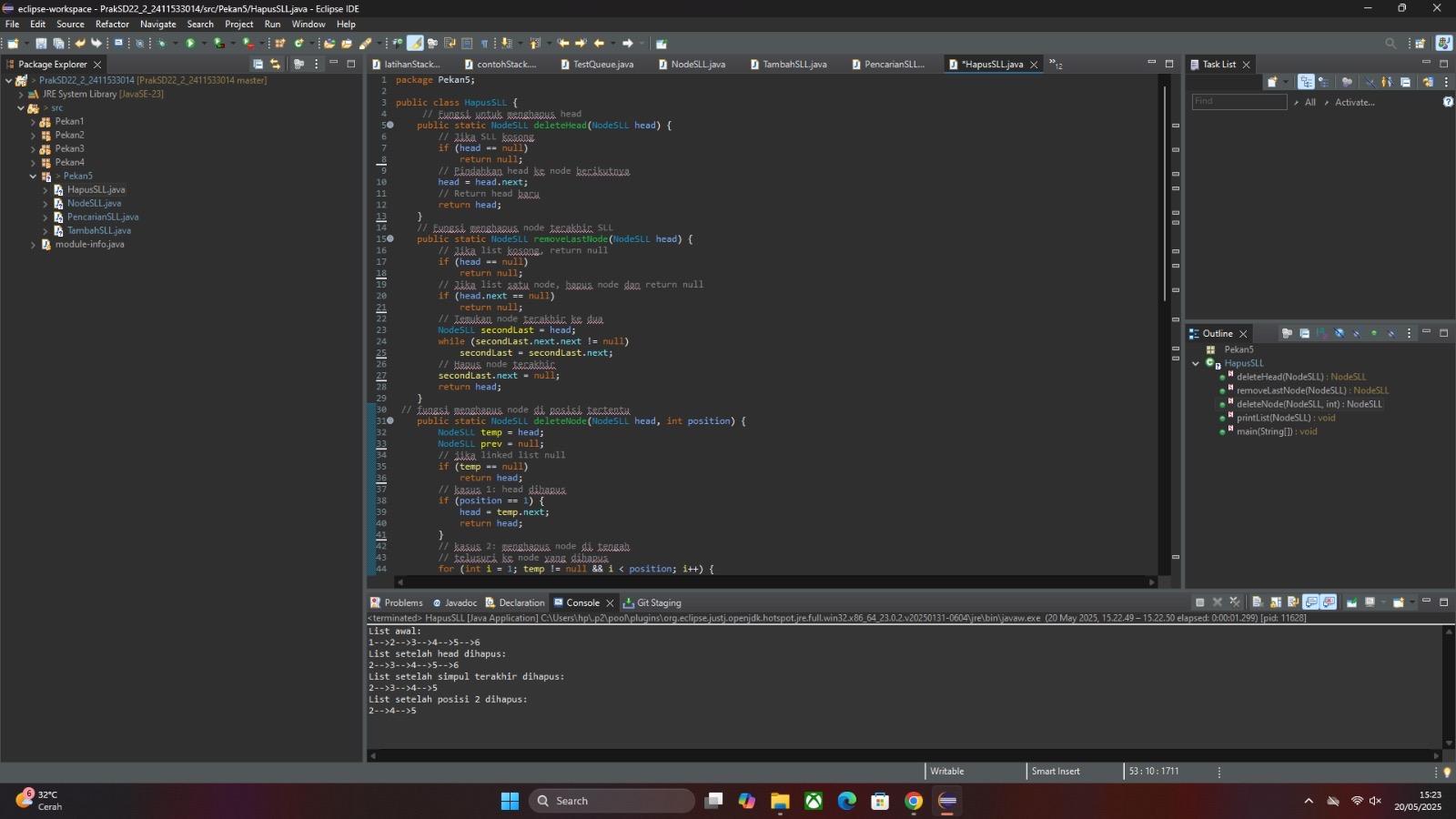
5. public NodeSLL(int data) {: Ini adalah konstruktor kelas NodeSLL. Konstruktor dipanggil saat objek NodeSLL baru dibuat. Argumen int data menerima nilai integer yang akan disimpan dalam node.

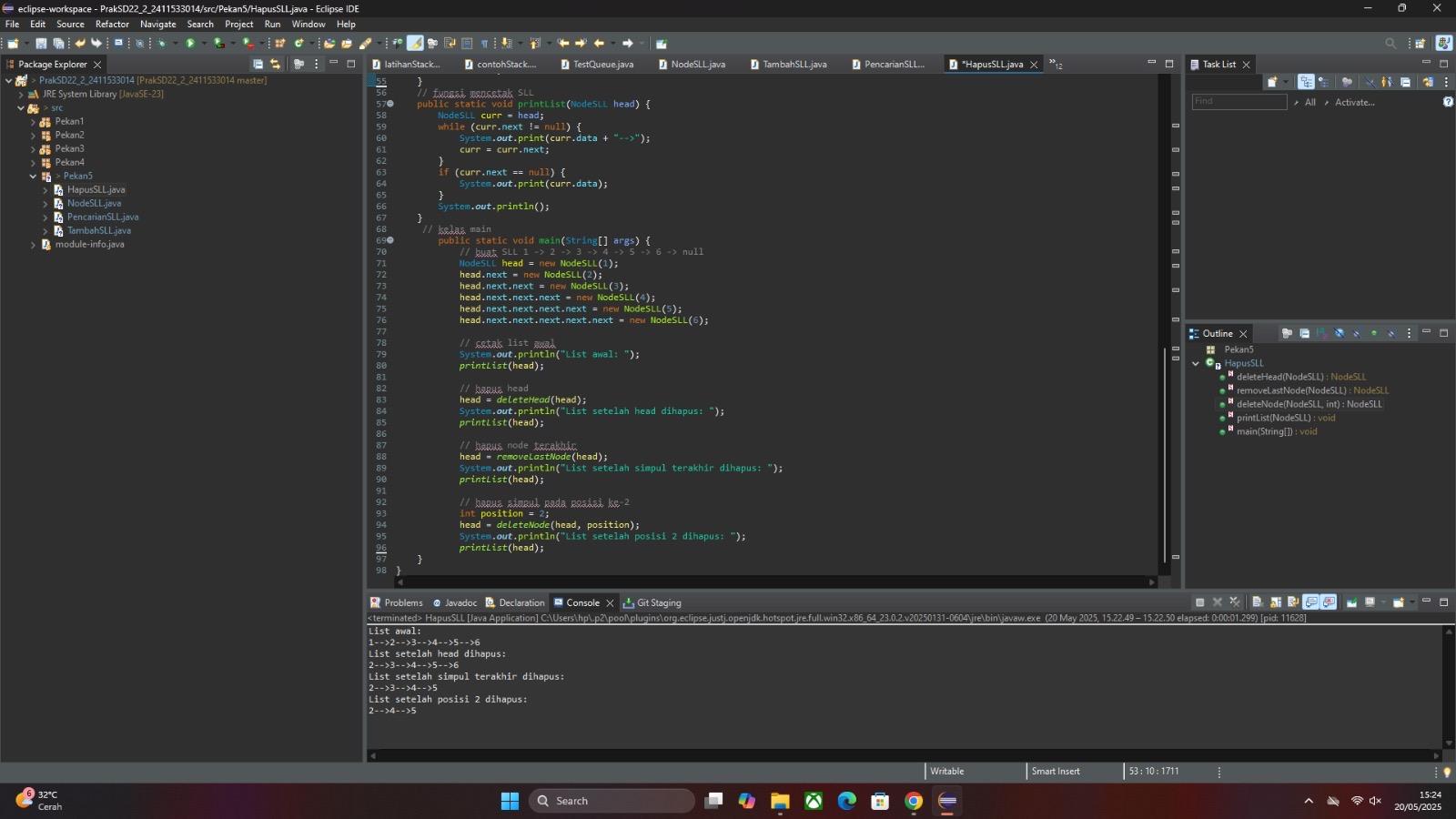
6. this.data = data;: Baris ini mengisi nilai variabel instance data dengan nilai yang diberikan sebagai argumen konstruktor. this merujuk ke objek NodeSLL yang sedang dibuat.

7. this.next = null;: Baris ini menetapkan variabel instance next menjadi null. Ini memastikan bahwa node baru yang dibuat awalnya tidak terhubung ke node lain.

8. }: Tanda kurung kurawal penutup untuk kelas NodeSLL.

1. HapusSLL





Langkah 1: Definisi Kelas Node

Kelas Node merepresentasikan setiap simpul dalam linked list. Setiap simpul menyimpan data dan referensi ke simpul berikutnya.

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

Langkah 2: Definisi Kelas SinglyLinkedList

Kelas ini akan mengelola linked list dan menyediakan fungsi-fungsi untuk manipulasi.

class SinglyLinkedList {

private Node head; // Referensi ke kepala list

public SinglyLinkedList() {

head = null; // List kosong pada awalnya

}

// Menambahkan node di awal

public void addFirst(int data) {

Node newNode = new Node(data);

newNode.next = head;

head = newNode;

}

// Menambahkan node di akhir

public void addLast(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

return;

}

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

// Menghapus node di awal

public void removeFirst() {

if (head == null) {

System.out.println("List kosong!"); // Penanganan kesalahan

return;

}

head = head.next;

}

// Menghapus node di akhir

public void removeLast() {

if (head == null) {

System.out.println("List kosong!"); // Penanganan kesalahan

return;

}

if (head.next == null) { // Hanya satu node

head = null;

return;

}

Node current = head;

while (current.next.next != null) {

current = current.next;

}

current.next = null;

}

// Menghapus node pada posisi tertentu (index berbasis 1)

public void removeAt(int index) {

if (head == null) {

System.out.println("List kosong!"); // Penanganan kesalahan

return;

}

if (index <= 0) {

System.out.println("Index tidak valid!"); // Penanganan kesalahan

return;

}

if (index == 1) {

removeFirst();

return;

}

Node current = head;

int count = 1;

while (current.next != null && count < index - 1) {

current = current.next;

count++;

}

if (current.next == null) {

System.out.println("Index di luar jangkauan!"); // Penanganan kesalahan

return;

}

current.next = current.next.next;

}

// Mencetak list

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

}

Langkah 3: Fungsi Utama (main)

Fungsi main akan mendemonstrasikan penggunaan kelas SinglyLinkedList.

public class Main {

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.addFirst(10);

list.addLast(20);

list.addFirst(5);

list.addLast(30);

System.out.print("List awal: ");

list.printList(); // Output: 5 10 20 30

list.removeFirst();

System.out.print("Setelah hapus awal: ");

list.printList(); // Output: 10 20 30

list.removeLast();

System.out.print("Setelah hapus akhir: ");

list.printList(); // Output: 10 20

list.removeAt(1); // Menghapus node pada index 1

System.out.print("Setelah hapus index 1: ");

list.printList(); // Output: 20

list.removeAt(5); //mencoba menghapus index diluar jangkauan

list.printList(); //output: 20

}

}

Langkah 4: Kompilasi dan Eksekusi

Simpan kode di atas dalam tiga file terpisah (Node.java, SinglyLinkedList.java, dan Main.java). Kompilasi dan jalankan menggunakan terminal:

javac Node.java SinglyLinkedList.java Main.java

java Main

1. KESIMPULAN

Pratikum ini berhasil mengimplementasikan operasi dasar pada singly linked list dalam bahasa Java. Implementasi kelas Node dan SinglyLinkedList, beserta fungsi-fungsi penambahan dan penghapusan node, menunjukkan pemahaman yang baik tentang struktur data ini. Pengujian yang dilakukan menunjukkan bahwa fungsi-fungsi tersebut bekerja dengan benar dan efisien, termasuk penanganan kondisi-kondisi khusus seperti list kosong atau indeks yang tidak valid. Keberhasilan implementasi ini membuktikan kemampuan untuk membangun dan memanipulasi struktur data dinamis, yang merupakan konsep penting dalam ilmu komputer. Pengembangan kode yang bersih dan terstruktur, disertai dengan penanganan kesalahan yang tepat, meningkatkan kualitas dan keandalan program.