**LAPORAN HASIL PRAKTIKUM ALSD**

**JOBSHEET 14 TREE**



Faiva Puspa Sahara

244107020036

TI – 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

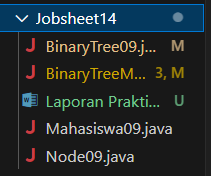
POLITEKNIK NEGERI MALANG

2025

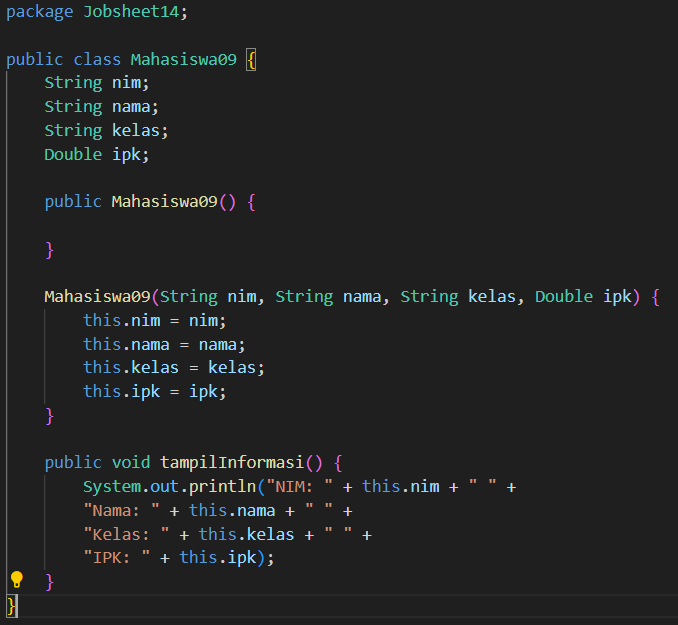
**HASIL PRAKTIKUM**

* 1. **Percobaan 1**

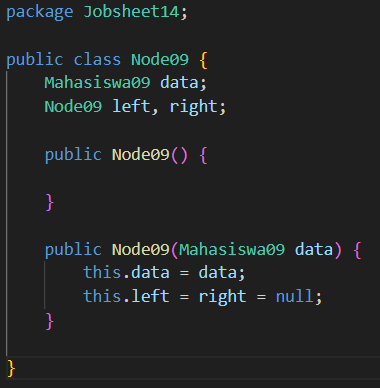
1. Buat folder baru bernama **Jobsheet14**. Tambahkan class-class berikut :



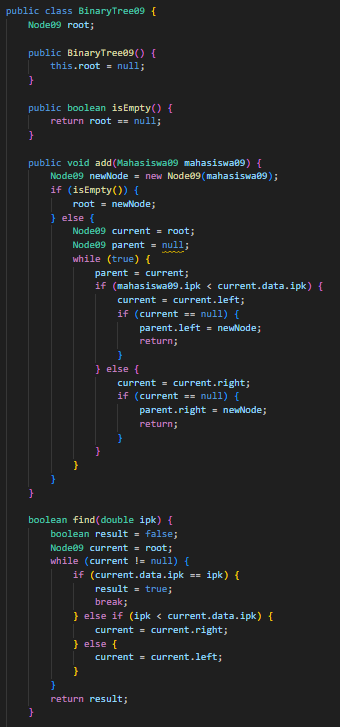
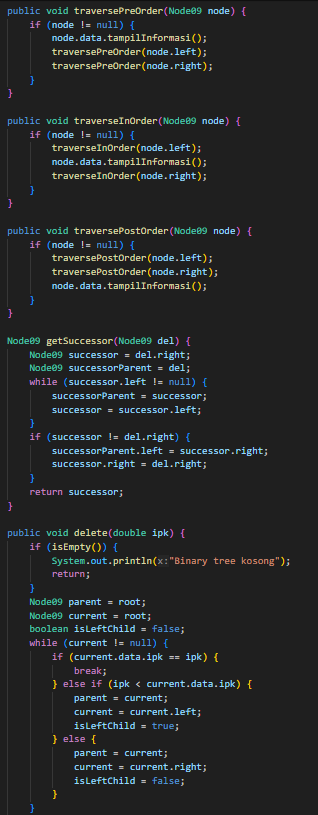
1. Di dalam class **Mahasiswa09**, deklarasikan atribut sesuai dengan diagram class Mahasiswa diatas. Tambahkan juga konstruktor dan method

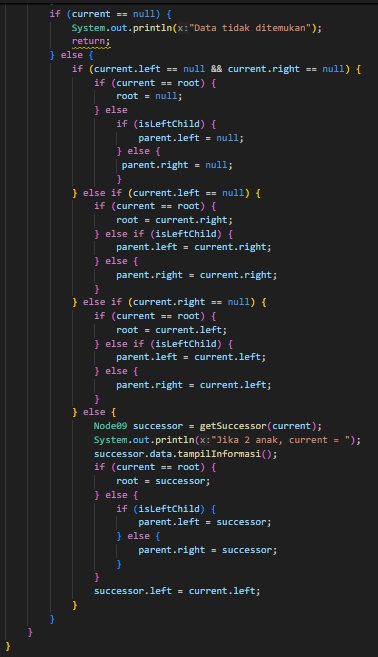


1. Di dalam class **Node09**, tambahkan atribut data, left dan right, serta konstruktor default dan berparameter

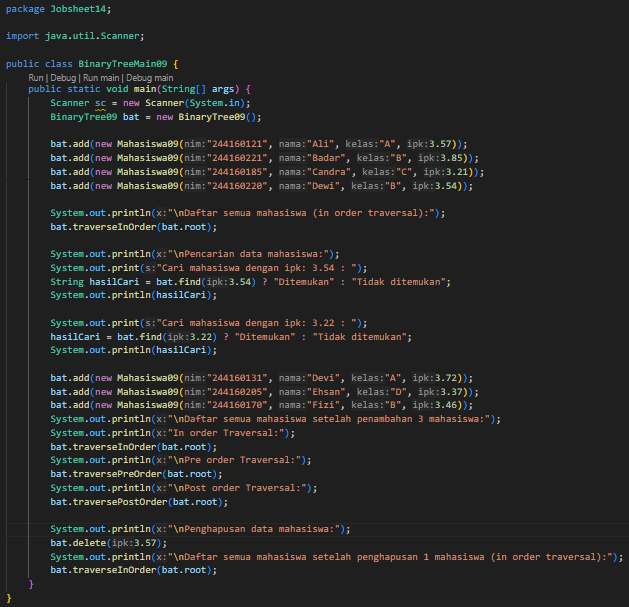


1. Di dalam class **BinaryTree09**, tambahkan atribut dan kode program sesuai dengan jobsheet 14

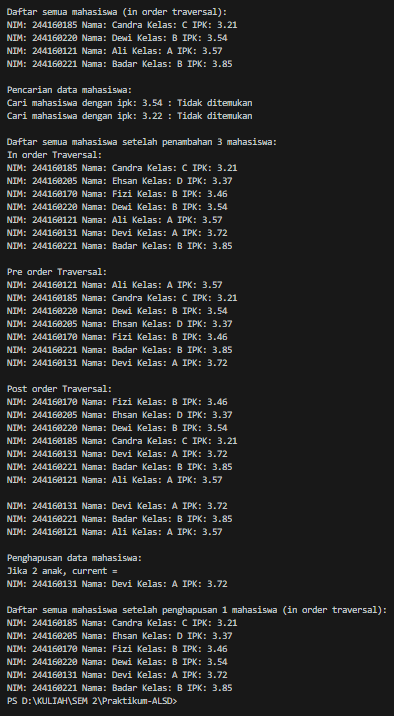
 



1. Buka class **BinaryTreeMain09** dan tambahkan method **main()** kemudian tambahkan kode sesuai pada jobsheet14



1. Run kode program



* **Pertanyaan**

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan disbanding binary tree biasa?

* Proses pencarian data dalam **binary search tree** lebih efektif karena struktur ini memiliki sifat di mana setiap node memiliki nilai lebih besar dari semua nilai di subtree kiri dan lebih kecil dari semua nilai di subtree kanan. Sifat ini untuk membuang setengah dari subtree yang tidak perlu diperiksa pada setiap langkah pencarian, sehingga mengurangi waktu pencarian dari O(n) (pada binary tree biasa) menjadi O(log n) pada kasus rata-rata dan terbaik (jika tree seimbang).

1. Untuk apakah di class Node, kegunaan dari atribut left dan right?

* Untuk menunjuk ke anak kiri dan anak kanan dari node tersebut. Pembentukan struktur pohon di mana setiap node dapat memiliki dua anak.

1. **a.** Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?

* Sebagai titik awal (node utama) untuk traversal, pencarian, penambahan, atau penghapusan node dalam binary tree.

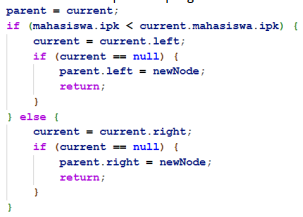
1. Ketika objek tree pertama kali dibuat, apakah nilai dari root?

* Nilai dari root adalah **null** karena belum ada node yang dimasukkan.

1. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?

* Ketika tree kosong, node pertama yang dimasukkan akan menjadi root. Proses root yang awalnya null akan diisi dengan node baru.
* 

1. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?



* Parent digunakan untuk menyimpan node sebelumnya sebelum current berpindah.
* Perbandingan **mahasiswa.ipk < current.data.ipk** menentukan apakah data baru harus ditempatkan di sebelah kiri atau kanan dari node saat ini.
* Jika posisi kosong ditemukan (**current == null**), maka node baru akan ditempatkan di posisi tersebut (sebagai anak kiri atau kanan dari parent).

1. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?

* Saat menghapus node dengan dua anak, node tersebut digantikan dengan node pengganti (successor), node dengan nilai terkecil di subtree kanan (anak paling kiri dari subtree kanan).
* Method getSuccessor() mencari node pengganti
* Setelah ditemukan, data successor menggantikan data node yang dihapus, lalu successor dihapus dari posisi aslinya (karena successor pasti memiliki maksimal satu anak).
  1. **Percobaan 2**

1. ..
2. Buatlah class **BinaryTreeArray09** dan **BinaryTreeArrayMain09**.



1. Buat atribut data dan idxLast di dalam class **BinaryTreeArray09**. Buat juga method populateData() dan traverseInOrder().

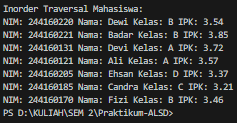
* Class BinaryTreeArray09



* Class BinaryTreeArrayMain09



1. Run program



* **Pertanyaan**

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray09**?

* Data, untuk menyimpan elemen-elemen dari binary tree dalam bentuk array.
* idxLast, untuk menunjukkan indeks dari elemen terakhir yang dimasukkan ke dalam binary tree.

1. Apakah kegunaan dari method **populateData**()?

* Untuk mengisi data binary tree ke dalam array. Biasanya digunakan untuk memberikan data awal atau membuat struktur pohon dari data yang sudah disiapkan.

1. Apakah kegunaan dari method **traverseInOrder**()?

* Untuk melakukan traversal in-order pada binary tree yang disimpan dalam array. Traversal in-order mengunjungi node dalam urutan (Traverse subtree kiri - node saat ini - Traverse subtree kanan.

1. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?

* Left Child, jika node disimpan di indeks i, maka posisi left child adalah 2\*i + 1. Jadi, jika node ada di indeks 2, left child ada di indeks 2\*2 + 1 = 5.
* Right Child, jika node disimpan di indeks i, maka posisi right child adalah 2\*i + 2. Jadi, jika node ada di indeks 2, right child ada di indeks 2\*2 + 2 = 6.
* Jadi, jika node binary tree disimpan dalam array pada indeks 2, posisi left child dan right child masing-masing adalah indeks 5 dan 6.

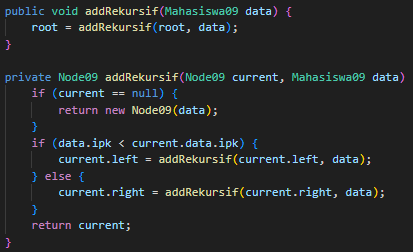
1. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?

* Untuk menandai indeks terakhir dari data yang sudah dimasukkan ke dalam array binary tree. Membatasi proses traversal dan memastikan indeks yang diakses berada dalam batas array yang valid.

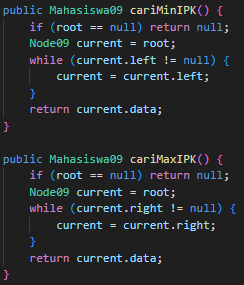
1. Mengapa indeks 2\*idxStart+1 dan 2\*idxStart+2 digunakan dalam pemanggilan rekursif, dan apa kaitannya dengan struktur pohon biner yang disusun dalam array?

* karena dalam representasi binary tree dalam array, anak kiri dari node di indeks i berada di 2\*i + 1. Dan anak kanan dari node di indeks i berada di 2\*i + 2.
* **Tugas Praktikum**

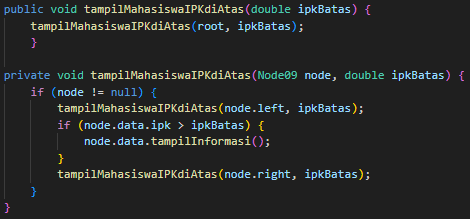
1. Buat method di dalam class **BinaryTree09** yang akan menambahkan node dengan cara rekursif (**addRekursif**()).



1. Buat method di dalam class **BinaryTree09** untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (**cariMinIPK**() dan **cariMaxIPK**()) yang ada di dalam binary search tree.



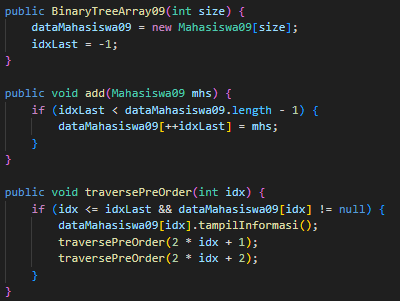
1. Buat method dalam class **BinaryTree09** untuk menampilkan data mahasiswa dengan IPK di atas suatu batas tertentu, misal di atas 3.50 (**tampilMahasiswaIPKdiAtas** (double ipkBatas)) yang ada di dalam binary search tree.



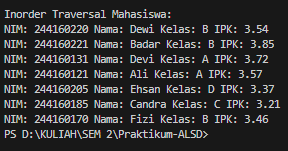
1. Modifikasi class BinaryTreeArray00 di atas, dan tambahkan :

• method add(Mahasiswa data) untuk memasukan data ke dalam binary tree

• method traversePreOrder()

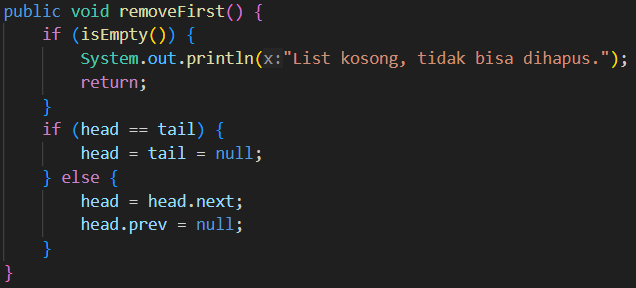


1. Run Kode Program

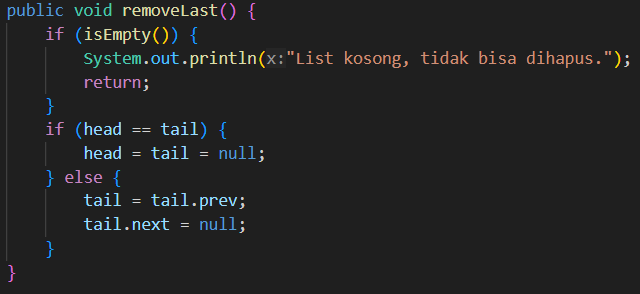


* 1. **Percobaan 2**

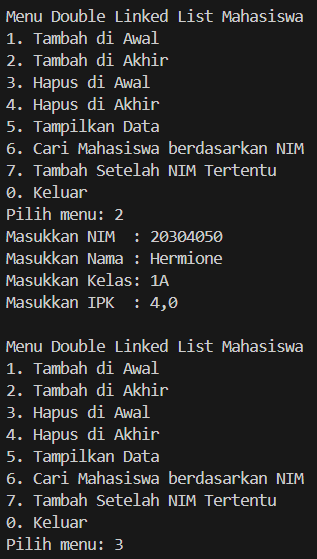
1. Buatlah method **removeFirst()** di dalam class **DoubleLinkedList09.java**

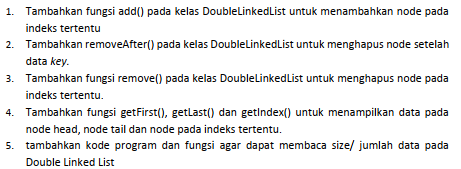
* 

1. Tambahkan method **removeLast()** di dalam class **DoubleLinkedList09.java**

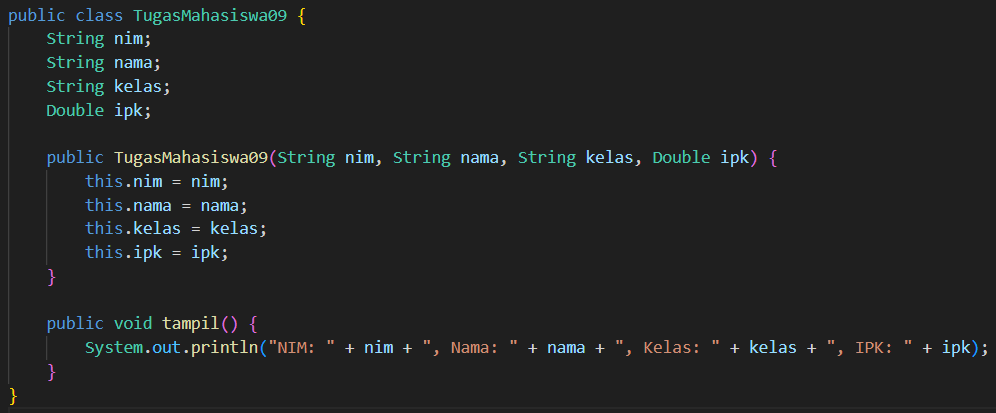
* 

1. Run Program

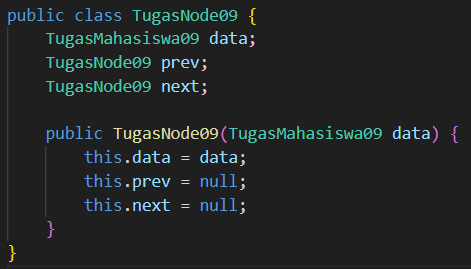
* 
  1. **Tugas**



* **Class TugasMahasiswa09.java**



* **Class TugasNode.java**



* **Class TugasDLL09.java**

public class TugasDLL09 {

TugasNode09 head;

TugasNode09 tail;

int size = 0;

public TugasDLL09() {

head = null;

tail = null;

size = 0;

}

public boolean isEmpty() {

return head == null;

}

public void addFirst(TugasMahasiswa09 data) {

TugasNode09 newNode = new TugasNode09(data);

if (isEmpty()) {

head = tail = newNode;

} else {

newNode.next = head;

head.prev = newNode;

head = newNode;

}

size++;

}

public void addLast(TugasMahasiswa09 data) {

TugasNode09 newNode = new TugasNode09(data);

if (isEmpty()) {

head = tail = newNode;

} else {

tail.next = newNode;

newNode.prev = tail;

tail = newNode;

}

size++;

}

// 1. Tambahkan fungsi add() untuk menambah node pada indeks tertentu

public void add(TugasMahasiswa09 data, int index) throws Exception {

if (index < 0 || index > size) throw new Exception("Indeks di luar batas!");

if (index == 0) {

addFirst(data);

} else if (index == size) {

addLast(data);

} else {

TugasNode09 newNode = new TugasNode09(data);

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

newNode.prev = current.prev;

newNode.next = current;

current.prev.next = newNode;

current.prev = newNode;

size++;

}

}

// 2. Tambahkan removeAfter() untuk menghapus node setelah key

public void removeAfter(String key) {

TugasNode09 current = head;

while (current != null && !current.data.nim.equals(key)) {

current = current.next;

}

if (current != null && current.next != null) {

TugasNode09 toDelete = current.next;

if (toDelete.next != null) {

current.next = toDelete.next;

toDelete.next.prev = current;

} else {

current.next = null;

tail = current;

}

size--;

}

}

// 3. Tambahkan fungsi remove() untuk menghapus node pada indeks tertentu

public void remove(int index) throws Exception {

if (isEmpty() || index < 0 || index >= size) throw new Exception("Indeks di luar batas!");

if (index == 0) {

removeFirst();

} else if (index == size - 1) {

removeLast();

} else {

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

current.prev.next = current.next;

current.next.prev = current.prev;

size--;

}

}

public void removeFirst() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

head = head.next;

head.prev = null;

}

size--;

}

public void removeLast() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

tail = tail.prev;

tail.next = null;

}

size--;

}

// 4. Fungsi getFirst(), getLast(), getAtIndex()

public TugasMahasiswa09 getFirst() {

if (isEmpty()) return null;

return head.data;

}

public TugasMahasiswa09 getLast() {

if (isEmpty()) return null;

return tail.data;

}

public TugasMahasiswa09 getAtIndex(int index) throws Exception {

if (isEmpty() || index < 0 || index >= size) throw new Exception("Indeks di luar batas!");

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

return current.data;

}

// 5. Fungsi untuk membaca jumlah data (size)

public int getSize() {

return size;

}

public void print() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

TugasNode09 current = head;

while (current != null) {

current.data.tampil();

current = current.next;

}

System.out.println("Jumlah data: " + size);

}

public TugasNode09 search(String nim) {

TugasNode09 current = head;

while (current != null) {

if (current.data.nim.equals(nim)) {

return current;

}

current = current.next;

}

return null;

}

}

head = newNode;

}

size++;

}

public void addLast(TugasMahasiswa09 data) {

TugasNode09 newNode = new TugasNode09(data);

if (isEmpty()) {

head = tail = newNode;

} else {

tail.next = newNode;

newNode.prev = tail;

tail = newNode;

}

size++;

}

// 1.

public void add(TugasMahasiswa09 data, int index) throws Exception {

if (index < 0 || index > size) throw new Exception("Indeks di luar batas!");

if (index == 0) {

addFirst(data);

} else if (index == size) {

addLast(data);

} else {

TugasNode09 newNode = new TugasNode09(data);

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

newNode.prev = current.prev;

newNode.next = current;

current.prev.next = newNode;

current.prev = newNode;

size++;

}

}

// 2.

public void removeAfter(String key) {

TugasNode09 current = head;

while (current != null && !current.data.nim.equals(key)) {

current = current.next;

}

if (current != null && current.next != null) {

TugasNode09 toDelete = current.next;

if (toDelete.next != null) {

current.next = toDelete.next;

toDelete.next.prev = current;

} else {

current.next = null;

tail = current;

}

size--;

}

}

// 3.

public void remove(int index) throws Exception {

if (isEmpty() || index < 0 || index >= size) throw new Exception("Indeks di luar batas!");

if (index == 0) {

removeFirst();

} else if (index == size - 1) {

removeLast();

} else {

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

current.prev.next = current.next;

current.next.prev = current.prev;

size--;

}

}

public void removeFirst() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

head = head.next;

head.prev = null;

}

size--;

}

public void removeLast() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

tail = tail.prev;

tail.next = null;

}

size--;

}

// 4. Fungsi getFirst(), getLast(), getAtIndex()

public TugasMahasiswa09 getFirst() {

if (isEmpty()) return null;

return head.data;

}

public TugasMahasiswa09 getLast() {

if (isEmpty()) return null;

return tail.data;

}

public TugasMahasiswa09 getAtIndex(int index) throws Exception {

if (isEmpty() || index < 0 || index >= size) throw new Exception("Indeks di luar batas!");

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

return current.data;

}

// 5. Fungsi untuk membaca jumlah data (size)

public int getSize() {

return size;

}

public void print() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

TugasNode09 current = head;

while (current != null) {

current.data.tampil();

current = current.next;

}

System.out.println("Jumlah data: " + size);

}

public TugasNode09 search(String nim) {

TugasNode09 current = head;

while (current != null) {

if (current.data.nim.equals(nim)) {

return current;

}

current = current.next;

}

return null;

}

}

current = current.next;

}

current.prev.next = current.next;

current.next.prev = current.prev;

size--;

}

}

public void removeFirst() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

head = head.next;

head.prev = null;

}

size--;

}

public void removeLast() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

if (head == tail) {

head = tail = null;

} else {

tail = tail.prev;

tail.next = null;

}

size--;

}

// 4.

public TugasMahasiswa09 getFirst() {

if (isEmpty()) return null;

return head.data;

}

public TugasMahasiswa09 getLast() {

if (isEmpty()) return null;

return tail.data;

}

public TugasMahasiswa09 getAtIndex(int index) throws Exception {

if (isEmpty() || index < 0 || index >= size) throw new Exception("Indeks di luar batas!");

TugasNode09 current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

return current.data;

}

// 5.

public int getSize() {

return size;

}

public void print() {

if (isEmpty()) {

System.out.println("Linked list masih kosong.");

return;

}

TugasNode09 current = head;

while (current != null) {

current.data.tampil();

current = current.next;

}

System.out.println("Jumlah data: " + size);

}

public TugasNode09 search(String nim) {

TugasNode09 current = head;

while (current != null) {

if (current.data.nim.equals(nim)) {

return current;

}

current = current.next;

}

return null;

}

}

current = current.next;

}

System.out.println("Jumlah data: " + size);

}

public TugasNode09 search(String nim) {

TugasNode09 current = head;

while (current != null) {

if (current.data.nim.equals(nim)) {

return current;

}

current = current.next;

}

return null;

}

}

* **Class TugasDLLMain09.java**

import java.util.Scanner;

public class TugasDLLMain09 {

public static void main(String[] args) {

TugasDLL09 list = new TugasDLL09();

Scanner sc = new Scanner(System.in);

int pilihan;

do {

System.out.println("\nMenu Double Linked List Mahasiswa");

System.out.println("1. Tambah di Awal");

System.out.println("2. Tambah di Akhir");

System.out.println("3. Tambah di Indeks Tertentu");

System.out.println("4. Hapus di Awal");

System.out.println("5. Hapus di Akhir");

System.out.println("6. Hapus di Indeks Tertentu");

System.out.println("7. Hapus Setelah NIM Tertentu");

System.out.println("8. Tampilkan Data");

System.out.println("9. Tampilkan Data Head");

System.out.println("10. Tampilkan Data Tail");

System.out.println("11. Tampilkan Data pada Indeks Tertentu");

System.out.println("12. Tampilkan Jumlah Data");

System.out.println("0. Keluar");

System.out.print("Pilih menu: ");

pilihan = sc.nextInt();

sc.nextLine();

switch (pilihan) {

case 1 -> {

TugasMahasiswa09 mhs = inputMahasiswa(sc);

list.addFirst(mhs);

}

case 2 -> {

TugasMahasiswa09 mhs = inputMahasiswa(sc);

list.addLast(mhs);

}

case 3 -> {

try {

System.out.print("Masukkan indeks: ");

int idx = sc.nextInt();

sc.nextLine();

TugasMahasiswa09 mhs = inputMahasiswa(sc);

list.add(mhs, idx);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

case 4 -> list.removeFirst();

case 5 -> list.removeLast();

case 6 -> {

try {

System.out.print("Masukkan indeks: ");

int idx = sc.nextInt();

sc.nextLine();

list.remove(idx);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

case 7 -> {

System.out.print("Masukkan NIM yang akan dihapus setelahnya: ");

String nim = sc.nextLine();

list.removeAfter(nim);

}

case 8 -> list.print();

case 9 -> {

TugasMahasiswa09 mhs = list.getFirst();

if (mhs != null) mhs.tampil();

else System.out.println("Linked list masih kosong.");

}

case 10 -> {

TugasMahasiswa09 mhs = list.getLast();

if (mhs != null) mhs.tampil();

else System.out.println("Linked list masih kosong.");

}

case 11 -> {

try {

System.out.print("Masukkan indeks: ");

int idx = sc.nextInt();

sc.nextLine();

TugasMahasiswa09 mhs = list.getAtIndex(idx);

if (mhs != null) mhs.tampil();

else System.out.println("Data tidak ditemukan.");

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

case 12 -> System.out.println("Jumlah data: " + list.getSize());

case 0 -> System.out.println("Keluar dari program.");

default -> System.out.println("Pilihan tidak valid!");

}

} while (pilihan != 0);

sc.close();

}

public static TugasMahasiswa09 inputMahasiswa(Scanner sc) {

System.out.print("Masukkan NIM : ");

String nim = sc.nextLine();

System.out.print("Masukkan Nama : ");

String nama = sc.nextLine();

System.out.print("Masukkan Kelas: ");

String kelas = sc.nextLine();

System.out.print("Masukkan IPK : ");

Double ipk = sc.nextDouble();

sc.nextLine();

return new TugasMahasiswa09(nim, nama, kelas, ipk);

}

}

case 6 -> {

try {

System.out.print("Masukkan indeks: ");

int idx = sc.nextInt();

sc.nextLine();

list.remove(idx);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

case 7 -> {

System.out.print("Masukkan NIM yang akan dihapus setelahnya: ");

String nim = sc.nextLine();

list.removeAfter(nim);

}

case 8 -> list.print();

case 9 -> {

TugasMahasiswa09 mhs = list.getFirst();

if (mhs != null) mhs.tampil();

else System.out.println("Linked list masih kosong.");

}

case 10 -> {

TugasMahasiswa09 mhs = list.getLast();

if (mhs != null) mhs.tampil();

else System.out.println("Linked list masih kosong.");

}

case 11 -> {

try {

System.out.print("Masukkan indeks: ");

int idx = sc.nextInt();

sc.nextLine();

TugasMahasiswa09 mhs = list.getAtIndex(idx);

if (mhs != null) mhs.tampil();

else System.out.println("Data tidak ditemukan.");

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

case 12 -> System.out.println("Jumlah data: " + list.getSize());

case 0 -> System.out.println("Keluar dari program.");

default -> System.out.println("Pilihan tidak valid!");

}

} while (pilihan != 0);

sc.close();

}

public static TugasMahasiswa09 inputMahasiswa(Scanner sc) {

System.out.print("Masukkan NIM : ");

String nim = sc.nextLine();

System.out.print("Masukkan Nama : ");

String nama = sc.nextLine();

System.out.print("Masukkan Kelas: ");

String kelas = sc.nextLine();

System.out.print("Masukkan IPK : ");

Double ipk = sc.nextDouble();

sc.nextLine();

return new TugasMahasiswa09(nim, nama, kelas, ipk);

}

}

* Run program

