**LAPORAN HASIL PRAKTIKUM ALSD**

**JOBSHEET 12 DOUBLE LINKED LIST**



Faiva Puspa Sahara

244107020036

TI – 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

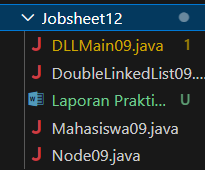
POLITEKNIK NEGERI MALANG

2025

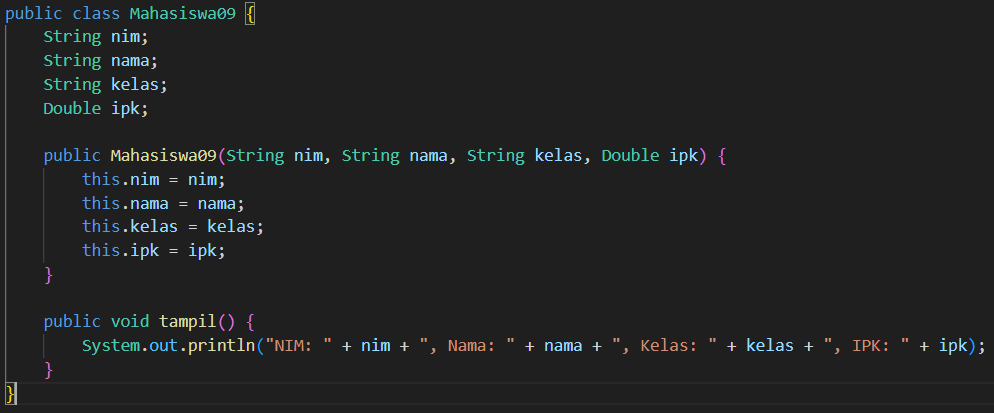
**HASIL PRAKTIKUM**

* 1. **Percobaan 1**

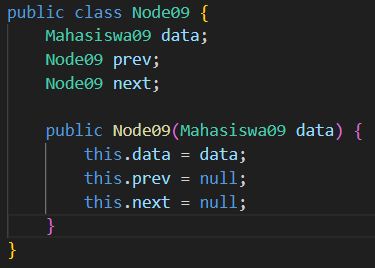
1. Buat folder baru bernama **Jobsheet12**. Buat file baru, dengan nama class **Mahasiswa09.java**



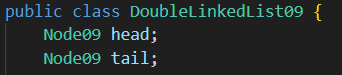
1. Deklarasikan atribut sesuai dengan diagram class **Mahasiswa09.java** di atas. Tambahkan juga konstruktor dan method



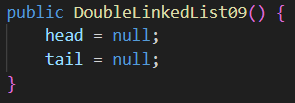
1. Buat class **Node09.java**



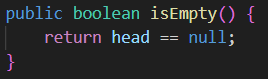
1. Buat class **DoubleLinkedList09.java** deklarasikan atribut sesuai dengan diagram class di atas.



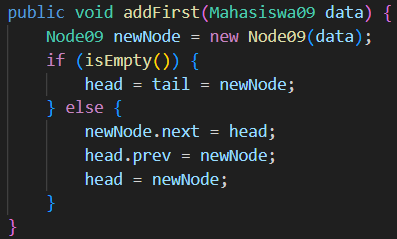
1. Buat konstruktor pada class **DoubleLinkedList09.java**



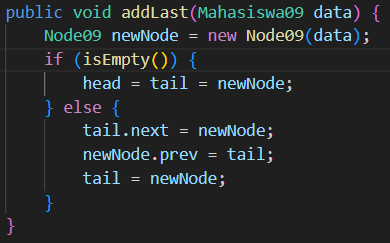
1. Buat method **isEmpty().** Digunakan untuk memastikan kondisi linked list kosong



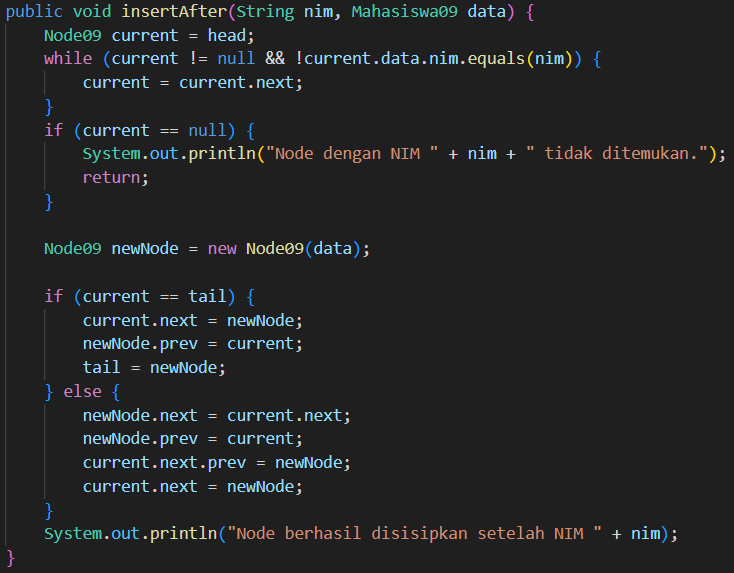
1. Buat method **addFirst().** Method ini akan menjalankan penambahan data dibagian depan linked list



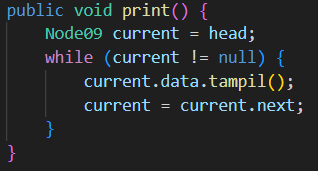
1. Method **addLast()** akan menambahkan data pada bagian belakang linked list



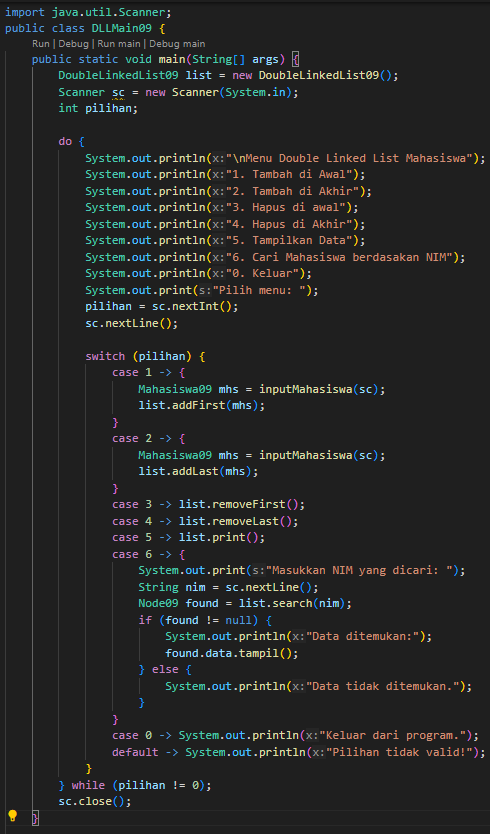
1. Untuk menambahkan data pada posisi setelah node yang menyimpan data *key,* dapat dibuat dengan cara sebagai berikut

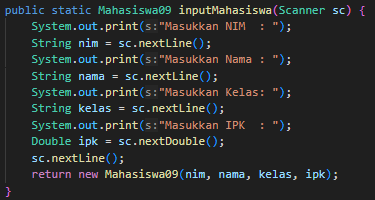


1. Untuk mencetak isi dari linked lists dibuat method **print().** Method ini akan mencetak isi linked list berapapun size-nya



1. Buatlah class **DLLMain09.java** untuk mengeksekusi semua method yang ada pada class **DoubleLinkedList09.java**





* **Pertanyaan**

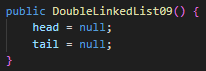
1. Jelaskan perbedaan antara single linked list dengan double linked list!

* Single linked list operasi satu arah dan hanya memiliki satu pointer pada setiap node, node berikutnya (next).
* Double linked list lebih fleksibel untuk operasi dua arah dan memiliki dua pointer pada setiap node, node berikutnya (next) dan ke node sebelumnya (prev).

1. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

* Atribut *next* pada class **Node01** digunakan untuk menunjuk node berikutnya dalam linked list, sedangkan atribut *prev* digunakan untuk menunjuk node sebelumnya.

1. Apa kegunaan dari konstruktor dibawah ini?



* Digunakan untuk menginisialisasi objek **DoubleLinkedList09** agar saat pertama kali dibuat, linked list masih kosong, yaitu dengan mengatur head dan tail bernilai null.

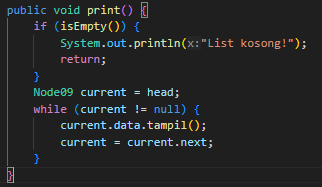
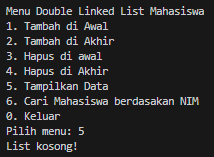
1. Apa maksud dari kode method **addFirst()**

* Berfungsi untuk menambahkan node baru di awal (head) linked list. Node baru akan menjadi head yang baru, dan jika list sebelumnya kosong, node tersebut juga menjadi **tail**. Jika list sudah berisi data, node baru akan menunjuk ke node head lama, dan head lama akan menunjuk balik ke node baru sebagai **prev**.

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

* Node yang sebelumnya menjadi **head** sekarang menunjuk kembali (**prev**) ke node baru ditambahkan di depan.

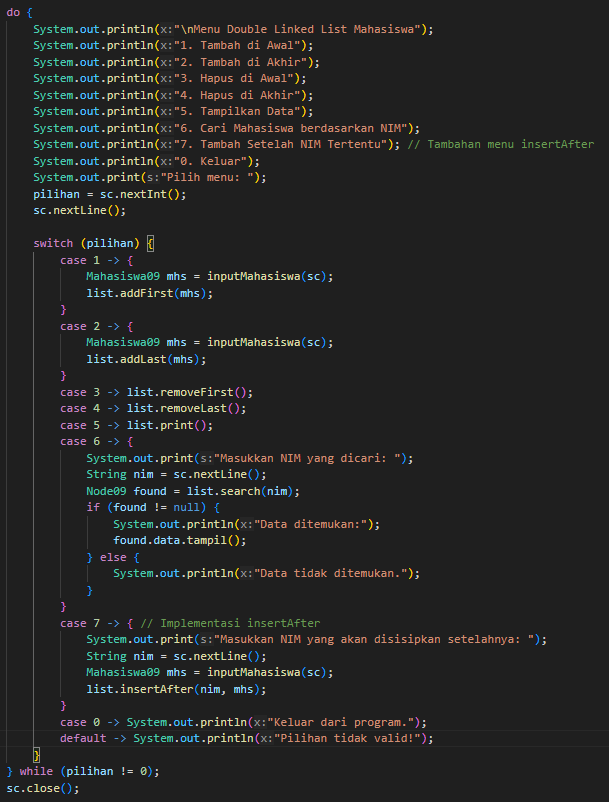
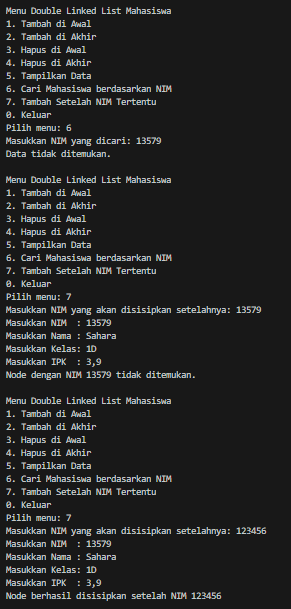
1. Modifikasi code pada fungsi **print()** agar dapat menampilkan warning/ pesan bahwa linked list masih dalam kondisi.

*  

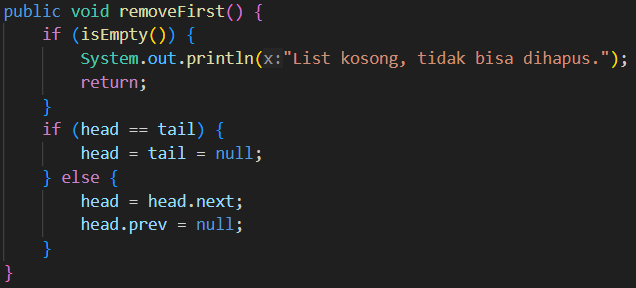
1. Pada **insertAfter()**, apa maksud dari kode ***current.next.prev = newNode;***

* Node yang setelah node current (*current.next*) akan mengupdate atribut *prev* agar menunjuk ke node baru yang baru saja disisipkan.

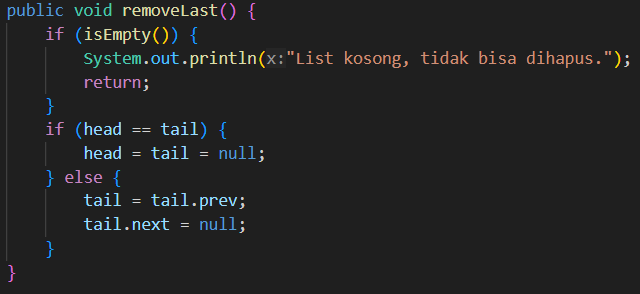
1. Modifikasi menu pilihan dan switch-case agar fungsi **insertAfter()** masuk ke dalam menu pilihan dan dapat berjalan dengan baik

*  
  1. **Percobaan 2**

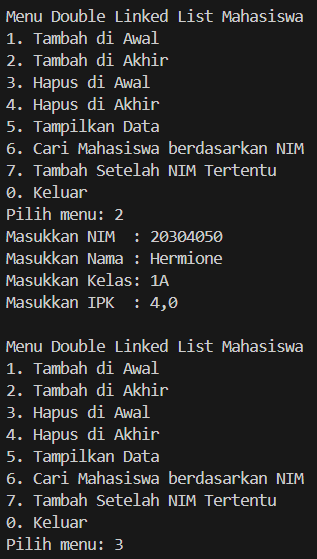
1. Buatlah method **removeFirst()** di dalam class **DoubleLinkedList09.java**

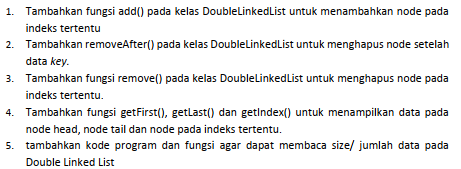
* 

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

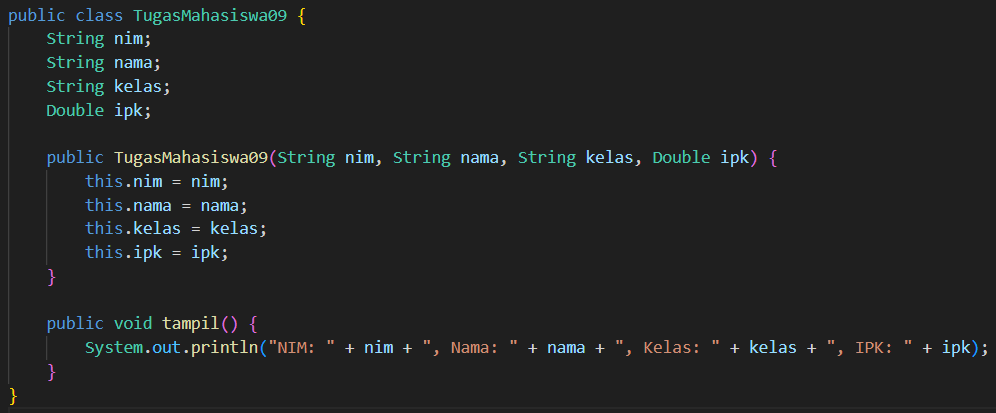
* 

1. Run Program

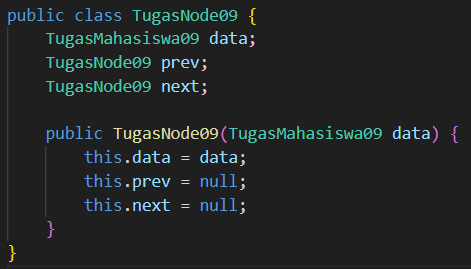
* 
  1. **Tugas**



* **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

