Laporan Hasil Praktikum

Algoritma Dan Struktur Data

Jobsheet 12



Angel Chelssa Leoniy Eka Permatasari

244107020202

1E

Program Studi Teknologi Informasi

Jurusan Teknik Informatika

POLINEMA

2025

**PRAKTIKUM 1**

**KODE PROGRAM**

class mahasiswa03.java

|  |
| --- |
| public class Mahasiswa03 {      public String nim;      public String nama;      public String kelas;      public double ipk;      public Mahasiswa03(String *nim*, String *nama*,String *kelas*, double *ipk*) {          this.nim = nim;          this.nama = nama;          this.kelas = kelas;          this.ipk = ipk;      }      public void Tampil() {          System.out.println("NIM: " + nim + " Nama: " +  nama + " Kelas: " + kelas + " IPK: " + ipk);      }  } |

class Node03.java

|  |
| --- |
| public class Node03 {      Mahasiswa03 data;      Node03 prev;      Node03 next;      public Node03(Node03 *prev*, Mahasiswa03 *data*, Node03 *next*) {          this.prev = prev;          this.data = data;          this.next = next;      }  } |

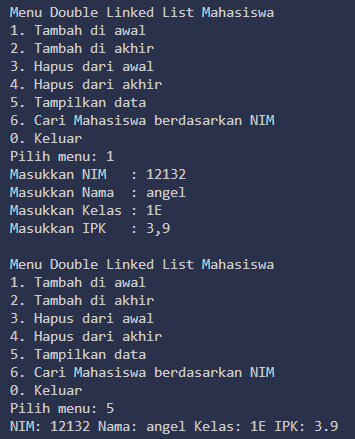
class DoubleLinkedList03.java

|  |
| --- |
| public class DoubleLinkedLists03 {      Node03 head;      Node03 tail;      public DoubleLinkedLists03() {          head = tail = null;      }      public boolean isEmpty() {          return head == null;      }      public void addFirst(Mahasiswa03 *data*) {          Node03 newNode = new Node03(null, data, head);          if (isEmpty()) {              head = tail = newNode;          } else {              head.prev = newNode;              head = newNode;          }      }      public void addLast(Mahasiswa03 *data*) {          Node03 newNode = new Node03(tail, data, null);          if (isEmpty()) {              head = tail = newNode;          } else {              tail.next = newNode;              tail = newNode;          }      }      public void removeFirst() {          if (isEmpty()) {              System.out.println("List kosong, tidak bisa menghapus.");          } else if (head == tail) {              System.out.println("Data yang dihapus:");              head.data.Tampil();              head = tail = null;          } else {              System.out.println("Data yang dihapus:");              head.data.Tampil();              head = head.next;              head.prev = null;          }      }      public void removeLast() {          if (isEmpty()) {              System.out.println("List kosong, tidak bisa menghapus.");          } else if (head == tail) {              System.out.println("Data yang dihapus:");              tail.data.Tampil();              head = tail = null;          } else {              System.out.println("Data yang dihapus:");              tail.data.Tampil();              tail = tail.prev;              tail.next = null;          }      }      public Node03 search(String *nim*) {          Node03 current = head;          while (current != null) {              if (current.data.nim.equals(nim)) {                  return current;              }              current = current.next;          }          return null;      }      public void print() {          if (isEmpty()) {              System.out.println("Linked List kosong.");          } else {              Node03 current = head;              while (current != null) {                  current.data.Tampil();                  current = current.next;              }          }      }  } |

class DoubleLinkedListMain03.java

|  |
| --- |
| import java.lang.classfile.components.ClassPrinter.Node;  import java.util.Scanner;  public class DoubleLinkedListMain03 {      public static void main(String[] *args*) {          DoubleLinkedLists03 list = new DoubleLinkedLists03();          Scanner scan = new Scanner(System.in);          int pilihan;          do {              System.out.println("=== Menu ===");              System.out.println("1. Tambah Data Mahasiswa di Awal");              System.out.println("2. Tambah Data Mahasiswa di Akhir");              System.out.println("3. Hapus dari awal");              System.out.println("4. Hapus dari akhir");              System.out.println("5. Tampilkan Data Mahasiswa");              System.out.println("6. cari Data Mahasiswa");              System.out.println("0. Keluar");              System.out.print("Pilih menu: ");              pilihan = scan.nextInt();              scan.nextLine(); *// Clear the newline character*              switch (pilihan) {                  case 1 -> {                      Mahasiswa03 mhs = inputMahasiswa(scan);                      list.addFirst(mhs);                  }                  case 2 -> {                      Mahasiswa03 mhs = inputMahasiswa(scan);                      list.addLast(mhs);                  }                  case 3 -> list.removeFirst();                  case 4 -> list.removeLast();                  case 5 -> list.print();                  case 6 -> {                      System.out.print("Masukkan NIM Mahasiswa yang dicari: ");                      String nim = scan.nextLine();                      Node03 found = list.search(nim);                      if (found != null) {                          System.out.println("Data Mahasiswa ditemukan:");                          found.data.Tampil();                      } else {                          System.out.println("data tidak ditemukan");                      }                  }                  case 0 -> System.out.println("Keluar dari program.");                  default -> System.out.println("pilihan tidak valid");              }          }          while (pilihan != 0);          scan.close();      }  } |

**HASIL KODE**



**PERTANYAAN**

1. Jelaskan perbedaan antara Single Linked List dengan Double Linked Lists!

Jawab: Single itu tiap node cuma punya 1 arah, dia tahu siapa yang di depan (next), tapi nggak tahu siapa yang di belakang.

Double itu lebih keren, bisa maju dan mundur, karena dia punya next dan prev. Jadi fleksibel kalau mau nambah atau hapus data di tengah-tengah.

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

Jawab: next itu buat nunjuk ke node selanjutnya.

prev itu buat ngarah ke node sebelumnya.

Jadi fungsinya biar kita bisa jalan ke depan dan ke belakang di list-nya. Biar nggak cuma satu arah doang.

1. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?

Jawab: Biar pas list pertama kali dibuat, isi list-nya kosong.

Jadi head dan tail diset jadi null dulu, nanti baru diisi kalau kita nambah data.

1. Pada method addFirst(), apa maksud dari kode berikut?

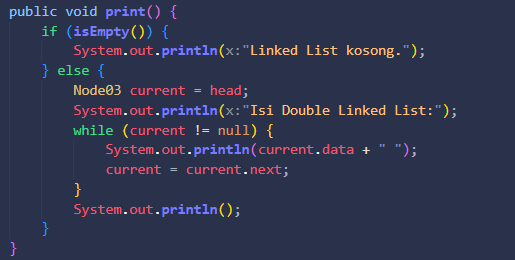
Jawab: Nyambungin node baru ke depan list, terus jadiin dia head.

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

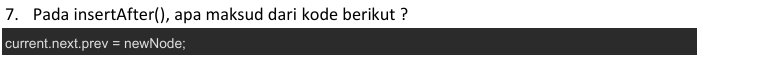
Jawab: Biar node lama tahu node barunya ada di depannya.

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

Jawab:



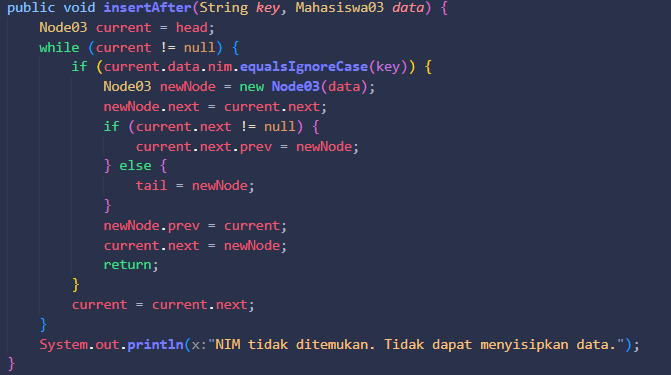
1. Soal



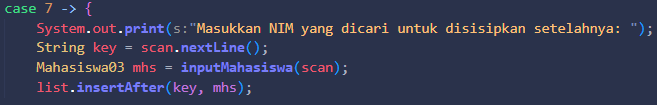
Jawab: Kode current.next.prev = newNode; berfungsi menghubungkan node setelah current ke newNode lewat pointer prev, agar hubungan dua arah pada double linked list tetap terjaga.

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

Jawab:

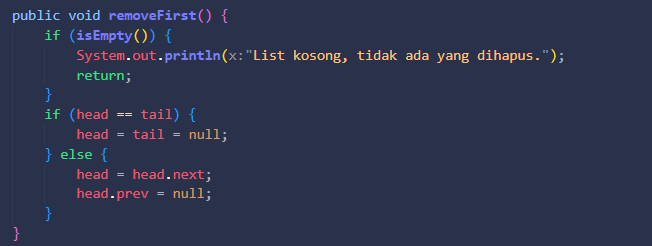




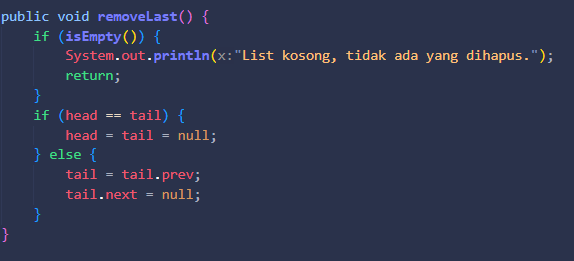


**PRAKTIKUM 2**

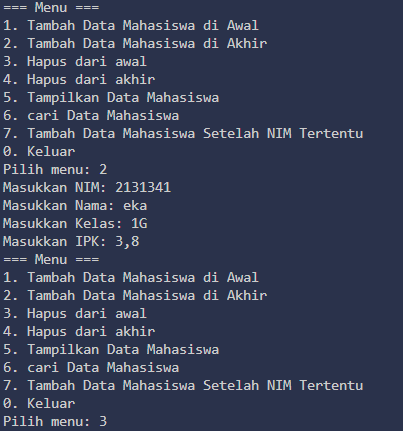
1. Buat method removeFirst()



1. Buat method removeLast()

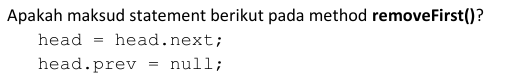


**HASIL KODE**



**PERTANYAAN**

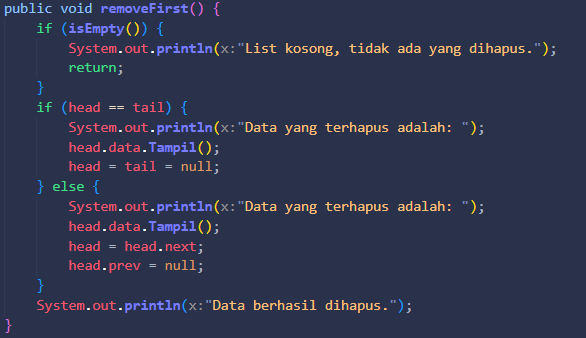
1. Soal



Jawab: Dua baris ini digunakan untuk menghapus node pertama pada double linked list dengan menggeser head ke node berikutnya dan memutus koneksi ke node lama.

1. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “

Jawab: modifikasi



**TUGAS**

**KODE PROGRAM**

Class Mahasiswa03.java

|  |
| --- |
| public class Mahasiswa03 {      public String nim;      public String nama;      public String kelas;      public double ipk;      public Mahasiswa03(String nim, String nama,String kelas, double ipk) {          this.nim = nim;          this.nama = nama;          this.kelas = kelas;          this.ipk = ipk;      }      public void Tampil() {          System.out.println("NIM: " + nim + " Nama: " +  nama + " Kelas: " + kelas + " IPK: " + ipk);      }  } |

Class Node03.java

|  |
| --- |
| public class Node03 {      Mahasiswa03 data;      Node03 prev;      Node03 next;     public Node03(Mahasiswa03 *data*) {          this.data = data;          this.prev = null;          this.next = null;      }  } |

Class DoubleLinkedList03.java

|  |
| --- |
| import java.lang.classfile.components.ClassPrinter.Node;  public class DoubleLinkedLists03 {      Node03 head;      Node03 tail;      public DoubleLinkedLists03() {          head = null;          tail = null;      }      public boolean isEmpty() {          return head == null;      }      public void addFirst(Mahasiswa03 *data*) {          Node03 newNode = new Node03(data);          if (isEmpty()) {              head = newNode;              tail = newNode;          } else {              newNode.next = head;              head.prev = newNode;              head = newNode;          }      }      public void addLast(Mahasiswa03 *data*) {          Node03 newNode = new Node03(data);          if (isEmpty()) {              head = newNode;              tail = newNode;          } else {              tail.next = newNode;              newNode.prev = tail;              tail = newNode;          }      }      public void insertAfter(String *key*, Mahasiswa03 *data*) {          Node03 current = head;          while (current != null) {              if (current.data.nim.equalsIgnoreCase(key)) {                  Node03 newNode = new Node03(data);                  newNode.next = current.next;                  if (current.next != null) {                      current.next.prev = newNode;                  } else {                      tail = newNode;                  }                  newNode.prev = current;                  current.next = newNode;                  return;              }              current = current.next;          }          System.out.println("NIM tidak ditemukan. Tidak dapat menyisipkan data.");      }      public void print() {          Node03 current = head;          while (current != null) {              current.data.Tampil();              current = current.next;          }      }      public void removeFirst() {          if (isEmpty()) {              System.out.println("List kosong, tidak ada yang dihapus.");              return;          }          if (head == tail) {              System.out.println("Data yang terhapus adalah: ");              head.data.Tampil();              head = tail = null;          } else {              System.out.println("Data yang terhapus adalah: ");              head.data.Tampil();              head = head.next;              head.prev = null;          }          System.out.println("Data berhasil dihapus.");      }      public void removeLast() {          if (isEmpty()) {              System.out.println("List kosong, tidak ada yang dihapus.");              return;          }          if (head == tail) {              head = tail = null;          } else {              tail = tail.prev;              tail.next = null;          }      }      public Node03 search(String *nim*) {          Node03 current = head;          while (current != null) {              if (current.data.nim.equalsIgnoreCase(nim)) {                  return current;              }              current = current.next;          }          return null;      }      public void add(int *index*, Mahasiswa03 *data*) {          if (index < 0 || index > (isEmpty() ? 0 : size())) {              System.out.println("Index tidak valid.");              return;          }          if (index == 0) {              addFirst(data);              return;          }          if (index == size()) {              addLast(data);              return;          }          Node03 current = head;          for(int i = 0; i < index - 1; i++) {              current = current.next;          }          Node03 newNode = new Node03(data);          newNode.prev = current.prev;          newNode.next = current;          current.prev.next = newNode;          current.prev = newNode;      }        public void removeAfter(String *key*) {          Node03 current = head;          while (current != null) {              if (current.data.nim.equalsIgnoreCase(key)) {                  if (current.next != null) {                      Node03 toDelete = current.next;                      current.next = toDelete.next;                      if (toDelete.next != null) {                          toDelete.next.prev = current;                      } else {                          tail = current;                      }                      System.out.println("Data setelah NIM " + key + " berhasil dihapus:");                      toDelete.data.Tampil();                      return;                  } else {                      System.out.println("Tidak ada data setelah NIM " + key);                      return;                  }              }              current = current.next;          }          System.out.println("NIM tidak ditemukan.");      }       public void remove(int *index*) {          if (isEmpty() || index < 0 || index >= size()) {              System.out.println("Indeks tidak valid atau list kosong.");              return;          }          if (index == 0) {              removeFirst();              return;          }          if (index == size() - 1) {              removeLast();              return;          }          Node03 current = head;          for (int i = 0; i < index; i++) {              current = current.next;          }          current.prev.next = current.next;          current.next.prev = current.prev;          System.out.println("Data yang dihapus:");          current.data.Tampil();      }      public void getFirst() {          if (!isEmpty()) {              System.out.println("Data pertama:");              head.data.Tampil();          } else {              System.out.println("List kosong.");          }      }      public void getLast() {          if (!isEmpty()) {              System.out.println("Data terakhir:");              tail.data.Tampil();          } else {              System.out.println("List kosong.");          }      }      public void getIndex(int *index*) {          if (index < 0 || index >= size()) {              System.out.println("Indeks tidak valid.");              return;          }          Node03 current = head;          for (int i = 0; i < index; i++) {              current = current.next;          }          System.out.println("Data pada indeks ke-" + index + ":");          current.data.Tampil();      }      public int size() {          int count = 0;          Node03 current = head;          while (current != null) {              count++;              current = current.next;          }          return count;      }    } |

Class DoubleLinkedListMain03.java

|  |
| --- |
| import java.lang.classfile.components.ClassPrinter.Node;  import java.util.Scanner;  public class DoubleLinkedListMain03 {      public static void main(String[] *args*) {          DoubleLinkedLists03 list = new DoubleLinkedLists03();          Scanner scan = new Scanner(System.in);          int pilihan;            do {              System.out.println("=== Menu ===");              System.out.println("1. Tambah Data Mahasiswa di Awal");              System.out.println("2. Tambah Data Mahasiswa di Akhir");              System.out.println("3. Hapus dari awal");              System.out.println("4. Hapus dari akhir");              System.out.println("5. Tampilkan Data Mahasiswa");              System.out.println("6. cari Data Mahasiswa");              System.out.println("7. Tambah Data Mahasiswa Setelah NIM Tertentu");              System.out.println("0. Keluar");              System.out.print("Pilih menu: ");              pilihan = scan.nextInt();              scan.nextLine(); *// Clear the newline character*              switch (pilihan) {                  case 1 -> {                      Mahasiswa03 mhs = inputMahasiswa(scan);                      list.addFirst(mhs);                  }                  case 2 -> {                      Mahasiswa03 mhs = inputMahasiswa(scan);                      list.addLast(mhs);                  }                  case 3 -> list.removeFirst();                  case 4 -> list.removeLast();                  case 5 -> list.print();                  case 6 -> {                      System.out.print("Masukkan NIM Mahasiswa yang dicari: ");                      String nim = scan.nextLine();                      Node03 found = list.search(nim);                      if (found != null) {                          System.out.println("Data Mahasiswa ditemukan:");                          found.data.Tampil();                      } else {                          System.out.println("data tidak ditemukan");                      }                  }                  case 7 -> {                      System.out.print("Masukkan NIM yang dicari untuk disisipkan setelahnya: ");                      String key = scan.nextLine();                      Mahasiswa03 mhs = inputMahasiswa(scan);                      list.insertAfter(key, mhs);              }                  case 0 -> System.out.println("Keluar dari program.");                  default -> System.out.println("pilihan tidak valid");              }          }          while (pilihan != 0);          scan.close();      }      public static Mahasiswa03 inputMahasiswa(Scanner *scan*) {          System.out.print("Masukkan NIM: ");          String nim = scan.nextLine();          System.out.print("Masukkan Nama: ");          String nama = scan.nextLine();          System.out.print("Masukkan Kelas: ");          String kelas = scan.nextLine();          System.out.print("Masukkan IPK: ");          double ipk = scan.nextDouble();          scan.nextLine();          return new Mahasiswa03(nim, nama, kelas, ipk);      }  } |

**HASIL KODE**

