LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

JOBSHEET 11



RIFO ANGGI BARBARA DANUARTA

244107020063

TI\_1E

PROGRAM STUDI D\_IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Percobaan 1

1. Pada Project yang sudah dibuat pada Minggu sebelumnya, buat package dengan nama Pertemuan12
2. Tambahkan class-class berikut:

a. Mahasiswa00.java

b. Node00.java

c. SingleLinkedList00.java

d. SLLMain00.java

Ganti 00 dengan nomer Absen Anda

1. Implementasikan Class Mahasiswa00 sesuai dengan diagram class berikut ini :
2. Implementasi class Node seperti gambar berikut ini

public class NodeMahasiswa26 {

Mahasiswa26 data;

NodeMahasiswa26 next;

public NodeMahasiswa26(Mahasiswa26 data, NodeMahasiswa26 next) {

this.data = data;

this.next = next;

}

}

1. Tambahkan attribute head dan tail pada class SingleLinkedList

NodeMahasiswa26 head;

NodeMahasiswa26 tail;

1. Sebagai langkah berikutnya, akan diimplementasikan method-method yang terdapat pada SingleLinkedList.
2. Tambahkan method isEmpty().

boolean isEmpty() {

return head == null;

}

1. Implementasi method untuk mencetak dengan menggunakan proses traverse.

public void print() {

if (!isEmpty()) {

NodeMahasiswa26 tmp = head;

System.out.println("Isi Linked List:\t");

while (tmp != null) {

tmp.data.tampilInformasi();

tmp = tmp.next;

}

System.out.println("");

} else {

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

}

}

1. Implementasikan method addFirst().

public void addFirst(Mahasiswa26 input) {

NodeMahasiswa26 nInput = new NodeMahasiswa26(input, null);

if (isEmpty()) {

head = nInput;

tail = nInput;

} else {

nInput.next = head;

head = nInput;

}

}

1. Implementasikan method addLast()

public void addLast(Mahasiswa26 input) {

NodeMahasiswa26 nInput = new NodeMahasiswa26(input, null);

if (isEmpty()) {

head = nInput;

tail = nInput;

} else {

tail.next = nInput;

tail = nInput;

}

}

1. Implementasikan method insertAfter, untuk memasukkan node yang memiliki data input setelah node yang memiliki data key.

public void insertAfter(String key, Mahasiswa26 input) {

NodeMahasiswa26 nInput = new NodeMahasiswa26(input, null);

NodeMahasiswa26 temp = head;

do {

if (temp.data.nama.equalsIgnoreCase(key)) {

nInput.next = temp.next;

temp.next = nInput;

if (nInput.next == null) {

tail = nInput;

}

break;

}

temp = temp.next;

} while (temp != null);

}

1. Tambahkan method penambahan node pada indeks tertentu.

public void insertAt(int index, Mahasiswa26 input) {

if (index < 0) {

System.out.println("indeks salah");

} else if (index == 0) {

addFirst(input);

} else {

NodeMahasiswa26 temp = head;

for (int i = 0; i < index - 1; i++) {

temp = temp.next;

}

temp.next = new NodeMahasiswa26(input, temp.next);

if (temp.next.next == null) {

tail = temp.next;

}

}

}

}

1. Pada class SLLMain00, buatlah fungsi main, kemudian buat object dari class SingleLinkedList.
2. Buat empat object mahasiswa dengan nama mhs1, mhs2, mhs3, mhs4 kemudian isi data setiap object melalui konstruktor.
3. Tambahkan Method penambahan data dan pencetakan data di setiap penambahannya agar terlihat perubahannya.

sll.print();

sll.addFirst(mhs4);

sll.print();

sll.addFirst(mhs1);

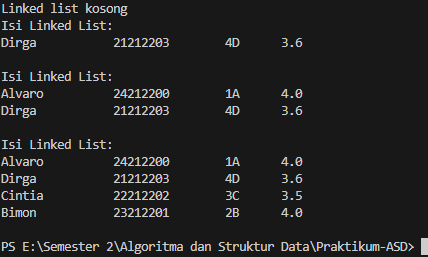
sll.print();

sll.insertAfter("Dirga", mhs3);

sll.insertAt(2, mhs2);

sll.print();

Verifikasi Hasil Percobaan



Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?
2. Jelaskan kegunaan variable temp secara umum pada setiap method!
3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

Jawaban

1. Karena saat mejalankan print objek linked list belum memiliki data\note satu pun makanya pemilihan if dengan kondisi negasi isempety tidak dipenuhi dan akan langsung ke else yang akan menampilkan linked list kosong.
2. Temp adalah pointer bantu/sementara untuk membantu note agar bisa menghapus data
3. Class SLLMain26

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

SingleLinkedList sll = new SingleLinkedList();

System.out.println("=== Tambah Data Mahasiswa ke Linked List ===");

System.out.print("Masukkan jumlah mahasiswa yang ingin ditambahkan: ");

int jumlah = sc.nextInt(); sc.nextLine();

for (int i = 1; i <= jumlah; i++) {

System.out.println("\nData Mahasiswa ke-" + i);

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

String nama = sc.nextLine();

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

String nim = sc.nextLine();

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

String kelas = sc.nextLine();

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

double ipk = sc.nextDouble(); sc.nextLine();

Mahasiswa26 mhs = new Mahasiswa26(nama, nim, kelas, ipk);

sll.addFirst(mhs);

}

System.out.println("\n=== Isi Linked List ===");

sll.print();

}

Percobaan 2

1. Implementasikan method untuk mengakses data dan indeks pada linked list
2. Tambahkan method untuk mendapatkan data pada indeks tertentu pada class Single Linked List

public void getData(int index) {

NodeMahasiswa26 tmp = head;

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

tmp = tmp.next;

}

tmp.data.tampilInformasi();

}

1. Implementasikan method indexOf.

public int indexOf(String key) {

NodeMahasiswa26 tmp = head;

int index = 0;

while (tmp != null && !tmp.data.nama.equalsIgnoreCase(key)) {

tmp = tmp.next;

index++;

}

if (tmp == null) {

return -1;

} else {

return index;

}

}

1. Tambahkan method removeFirst pada class SingleLinkedList

public void removeFirst() {

if (isEmpty()) {

System.out.println("Linked list masih kosong, tidak bisa dihapus");

} else if (head == tail) {

head = tail = null;

} else {

head = head.next;

}

}

1. Tambahkan method untuk menghapus data pada bagian belakang pada class SingleLinkedList

public void removeLast() {

if (isEmpty()) {

System.out.println("Linked list masih kosong, tidak bisa dihapus");

} else if (head == tail) {

head = tail = null;

} else {

NodeMahasiswa26 temp = head;

while (temp.next != tail) {

temp = temp.next;

}

temp.next = null;

tail = temp;

}

}

1. Sebagai langkah berikutnya, akan diimplementasikan method remove

public void remove(String key) {

if (isEmpty()) {

System.out.println("Linked list masih kosong, tidak bisa dihapus");

} else{

NodeMahasiswa26 temp = head;

while (temp != null) {

if ((temp.data.nama.equalsIgnoreCase(key)) && (temp == head)) {

this.removeFirst();

break;

} else if (temp.data.nama.equalsIgnoreCase(key)) {

temp.next = temp.next.next;

if (temp.next == null) {

tail = temp;

}

break;

}

temp = temp.next;

}

}

}

1. Implementasi method untuk menghapus node dengan menggunakan index.

public void removeAt(int index) {

if (index == 0) {

removeFirst();

} else {

NodeMahasiswa26 temp = head;

for (int i = 0; i < index - 1; i++) {

temp = temp.next;

}

temp.next = temp.next.next;

if (temp.next == null) {

tail = temp;

}

}

}

1. Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class SLLMain dengan menambahkan kode berikut

System.out.println("data index 1 : ");

sll.getData(1);

System.out.println("data mahasiswa an Bimon berada pada index : "+sll.indexOf("bimon"));

System.out.println();

sll.removeFirst();

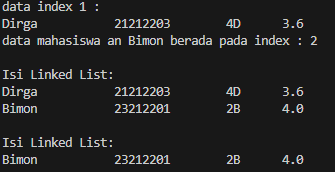
sll.removeLast();

sll.print();

sll.removeAt(0);

sll.print();

1. Jalankan class SLLMain



Pertanyaan

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!
2. Jelaskan kegunaan kode dibawah pada method remove

Jawaban

1. Karena untuk menghentikan perulangan while setelah data yang dicari berhasil ditemukan dan dihapus dari linked list.
2. Misalnya temp berada di node A, dan node berikutnya (A.next) adalah node B, yang berisi data mahasiswa yang ingin dihapus. Maka Dengan menulis A.next = B.next, temp melewati node B dari rantai linked list.Ini artinya node B tidak lagi dirujuk oleh node manapun, dan karena itu dianggap terhapus dari list.

Maksud pemilihan if

Menghapus node yang ada di tengah atau akhir linked list, dan memastikan pointer tail selalu menunjuk ke node terakhir yang benar setelah penghapusan dilakukan.

Tugas

package Jobsheet11.Tugas;

public class Mahasiswa26 {

String nama;

String nim;

String kelas;

public Mahasiswa26(String nama, String nim, String kelas) {

this.nama = nama;

this.nim = nim;

this.kelas = kelas;

}

public void tampilInformasi() {

System.out.println(nama + "\t\t" + nim + "\t" + kelas);

}

}

package Jobsheet11.Tugas;

public class NodeMahasiswa26 {

Mahasiswa26 data;

NodeMahasiswa26 next;

public NodeMahasiswa26(Mahasiswa26 data, NodeMahasiswa26 next) {

this.data = data;

this.next = next;

}

}

package Jobsheet11.Tugas;

public class SLL26 {

NodeMahasiswa26 front;

NodeMahasiswa26 rear;

int size;

int max;

public SLL26(int n) {

max = n;

front = rear = null;

size = 0;

}

public boolean isEmpty() {

return size == 0;

}

public boolean isFull() {

return size == max;

}

public void clear() {

if (!isEmpty()) {

front = rear = null;

size = 0;

System.out.println("Antrian berhasil dikosongkan.");

} else {

System.out.println("Antrian sudah kosong.");

}

}

public void tambahAntrian(Mahasiswa26 mhs) {

if (isFull()) {

System.out.println("Antrian penuh, tidak bisa menambah mahasiswa.");

return;

}

NodeMahasiswa26 newNode = new NodeMahasiswa26(mhs, null);

if (isEmpty()) {

front = rear = newNode;

} else {

rear.next = newNode;

rear = newNode;

}

size++;

System.out.println(mhs.nama + " berhasil masuk ke antrian.");

}

public Mahasiswa26 dequeue () {

if (isEmpty()) {

System.out.println("Antrian kosong.");

}

Mahasiswa26 mhs = front.data;

front = front.next;

if (front == null) {

rear = null;

}

size--;

System.out.println("Mahasiswa " + mhs.nama + " telah dipanggil.");

return mhs;

}

public void panggilAntrian() {

if (isEmpty()) {

System.out.println("Antrian kosong.");

} else {

Mahasiswa26 mhs = dequeue();

if (mhs != null) {

mhs.tampilInformasi();

}

}

}

public void tampilDepan() {

if (isEmpty()) {

System.out.println("Antrian kosong.");

} else {

System.out.println("Mahasiswa terdepan: ");

front.data.tampilInformasi();

}

}

public void tampilAkhir() {

if (isEmpty()) {

System.out.println("Antrian kosong.");

} else {

System.out.println("Mahasiswa terakhir: ");

rear.data.tampilInformasi();

}

}

public void tampilJumlahAntrian() {

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

}

}

package Jobsheet11.Tugas;

import java.util.Scanner;

public class Main26 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Masukkan jumlah mahasiswa: ");

int n = sc.nextInt(); sc.nextLine();

SLL26 linkedList = new SLL26(n);

while (true) {

System.out.println("===Sistem Antrian Unit Mahasiswa===");

System.out.println("1. Daftar Mahasiswa");

System.out.println("2. Mengosongkan Antrian");

System.out.println("3. Menambahkan Antrian");

System.out.println("4. Memanggil Antrian");

System.out.println("5. Menampilkan Antrian Terdepan");

System.out.println("6. Menampilkan Antrian Terakhir");

System.out.println("7. Menampilkan Jumlah Antrian");

System.out.println("8. Keluar");

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

int pilihan = sc.nextInt(); sc.nextLine();

switch (pilihan) {

case 1:

System.out.print("Masukkan nama mahasiswa: ");

String nama = sc.nextLine();

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

String nim = sc.nextLine();

System.out.print("Masukkan kelas mahasiswa: ");

String kelas = sc.nextLine();

Mahasiswa26 mhs = new Mahasiswa26(nama, nim, kelas);

linkedList.tambahAntrian(mhs);

case 2:

linkedList.clear();

break;

case 3:

linkedList.dequeue();

break;

case 4:

linkedList.panggilAntrian();

break;

case 5:

linkedList.tampilDepan();

break;

case 6:

linkedList.tampilAkhir();

break;

case 7:

linkedList.tampilJumlahAntrian();

break;

case 8:

System.out.println("Terima kasih telah menggunakan sistem antrian.");

System.exit(0);

default:

System.out.println("Pilihan tidak valid, silakan coba lagi.");

break;

}

}

}

}