Laporan Jobsheet 9

Algoritma dan Struktur Data

Disusun Oleh :

DHANISA PUTRI MASHILFA

NIM. 2341720212

TI-1E/07

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No. 9 Jatimulyo, Kecamatan Lowokwaru, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 6514

2.1 Pembuatan Single Linked List

2.1.1 Langkah - Langkah Praktikum Percobaan 2.1

a. Node.java

```
public class Node {

    int data;
    Node next;

   public Node(int nilai, Node Berikutnya) {
        data = nilai;
        next = Berikutnya;

    }
}
```

b. SingleLinkedList.java

```
public class SingleLinkedList {
    Node head, tail;

    boolean isEmpty() {
        return head != null;
    }

    void print() {
        if (isEmpty()) {
            Node tmp = head;
            System.out.print("Isi Linked List: ");
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println();
            System.out.println("Linked List kosong");
        }
    }

    void addFirst(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            //tail = ndInput;
            //head = ndInput;
            ndInput.next = head;
            head = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
            //ndInput.next = head;
            //head = ndInput;
        }
    }
```

```
    void addLast(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {

            tail.next = ndInput;
            tail = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int input) {
        Node ndInput = new Node(input, null);
        Node temp = head;
        do {
            if (temp.data == key) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next != null) {

                    tail = ndInput;
                    break;
                }
            }
            temp = temp.next;
        } while (temp != null);
    }

    void insertAt(int index, int input) {
        Node ndInput = new Node(input, null);
        if (index < 0) {
            System.out.println("Perbaiki logikanya!" + "kalau
indeksnya -1 bagaimana???");

        } else if (index == 0) {
            addFirst(input);
        } else {
            Node temp = head;
            for (int i = 1; i < index; i++) {
                temp = temp.next;
            }
            temp.next = new Node(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }
}
```

c. SLLMain.java

```
public class SLLMain {

```

```
    public static void main(String[] args) {
        SingleLinkedList singLL = new SingleLinkedList();

        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.insertAfter(700, 999);
        singLL.print();
        singLL.insertAt(3, 833);
        singLL.print();
    }

}
```

2.1.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
Linked list kosong
Isi Linked List:        890
Isi Linked List:        890     760
Isi Linked List:        700     890     760
Isi Linked List:        700     999     890     760
Isi Linked List:        700     999     890     833     760
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
1b
Linked List kosong
Isi Linked List: 890
Isi Linked List: 890    760
Isi Linked List: 700    890     760
Isi Linked List: 700    999     890     760
Isi Linked List: 700    999     890     833     760
PS C:\Users\asus\Documents\Semester2\Algoritma dan Struktur Data\Aljabar-Lini
er-dan-Sturktur-Data\Jobsheet9>
```

2.1.3 Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?
   → Karena implementasi metode isEmpty(), yang mengembalikan true jika head adalah null, pernyataan "Linked List Kosong" muncul dalam metode addFirst(int input). Tetapi dalam kondisi ini, ketika metode addFirst(int input) dipanggil, variabel ndInput dibuat sebagai node baru dengan nilai yang diberikan. Setelah itu, variabel head diatur ke ndInput. Ini menunjukkan bahwa daftar yang terhubung tidak lagi kosong karena memiliki setidaknya satu node yaitu, node yang ditambahkan pada awalnya.

2. Jelaskan kegunaan variable temp secara umum pada setiap method!
   → Pada setiap metode, variabel temp digunakan sebagai penanda untuk mengarahkan daftar terhubung dari awal hingga akhir dan mencari kondisi tertentu, seperti di mana untuk memasukkan elemen baru atau elemen yang memenuhi kriteria tertentu.

a. Metode print() : Variabel temp digunakan untuk melintasi seluruh linked list, dimulai dari head hingga akhir (null). Ini membantu mencetak nilai data dari setiap node dalam linked list.

b. Metode addFirst(int input) : Variabel temp tidak digunakan dalam metode ini karena hanya menambahkan elemen di awal linked list.

c. Metode addLast(int input) : Variabel temp tidak digunakan dalam metode ini karena hanya menambahkan elemen di akhir linked list.

d. Metode insertAfter(int key, int input) : Untuk menyisipkan elemen baru setelah node yang sesuai dengan kunci, variabel temp digunakan untuk melintasi daftar yang terhubung dan menemukan node yang memiliki nilai yang sama dengan kunci.

e. Metode insertAt(int index, int input) : Untuk menyisipkan elemen baru di posisi yang diinginkan dalam linked list, variabel temp digunakan untuk melintasi daftar terkait dan mencapai node sebelum indeks.

3. Perhatikan class **SingleLinkedList**, pada method **insertAt** Jelaskan kegunaan kode berikut

```
if(temp.next.next==null) tail=temp.next;
```

→ Dengan menggunakan kode if (temp.next.next == null), dapat mengetahui apakah temp adalah node kedua terakhir dalam daftar. Jika temp.next.next adalah null, itu berarti node terakhir dalam daftar adalah node setelah temp.next, yang merupakan node berikutnya dari temp.

2.2 Modifikasi Elemen pada Single Linked List

2.2.1 Langkah - Langkah Praktikum Percobaan 2.2

a. SingleLinkedList.java

```
public class SingleLinkedList {
    Node head, tail;

    boolean isEmpty() {
        return head != null;
    }

    void print() {
        if (isEmpty()) {
            Node tmp = head;
            System.out.print("Isi Linked List: ");
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println();
            System.out.println("Linked List kosong");
        }
    }
```

```
    void addFirst(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            //tail = ndInput;
            //head = ndInput;
            ndInput.next = head;
            head = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
            //ndInput.next = head;
            //head = ndInput;
        }
    }

    void addLast(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {

            tail.next = ndInput;
            tail = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int input) {
        Node ndInput = new Node(input, null);
        Node temp = head;
        do {
            if (temp.data == key) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next != null) {

                    tail = ndInput;
                    break;
                }
            }
            temp = temp.next;
        } while (temp != null);
    }

    void insertAt(int index, int input) {
        Node ndInput = new Node(input, null);
        if (index < 0) {
            System.out.println("Perbaiki logikanya!" + "kalau
indeksnya -1 bagaimana???");

        } else if (index == 0) {
            addFirst(input);
```

```
        } else {
            Node temp = head;
            for (int i = 1; i < index; i++) {
                temp = temp.next;
            }
            temp.next = new Node(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }

    int getData(int index) {
        Node tmp = head;
        for (int i = 0; i < index; i++) {
            if (tmp == null) {
                throw new IndexOutOfBoundsException("");
            }
            tmp = tmp.next;
        }
        if (tmp == null) {
            throw new IndexOutOfBoundsException("");
        }
        return tmp.data;
    }

    int indexOf(int key) {

        Node tmp = head;
        int index = 0;
        while(tmp != null && tmp.data != key) {
            tmp = tmp.next;
            index++;
        }
        if(tmp != null) {
            return index;
        } else {
            return -1;
        }
    }

    void removeFirst(){
        if(!isEmpty()) {
            System.out.println("Linked list masih kosong," + " tidak
dapat dihapus");
        } else if(head == tail){
            head = tail = null;
        } else {
            head = head.next;
        }
    }

    void removeLast() {
```

```
        if (!isEmpty()) {
            System.out.println("Linked list masih kosong, " + "tidak
dapat dihapus");
        //} else if(head != tail){
            //head = tail = null;
        } else {
            Node temp = head;
            while (temp.next.next != null) {
                temp = temp.next;
            }
            temp.next = null;
            tail = temp.next;
        }
    }

    void remove(int key) {
        if (!isEmpty()) {
            System.out.println("Linked list masih kosong, " + "tidak
dapat dihapus");
        } else {
            Node temp = head;
            while (temp != null) {
                // if (temp.data != key && temp == head) {
                //     removeFirst();
                //     break;
                // }
                if (temp.next.data == key) {
                    temp.next = temp.next.next;
                    if (temp.next == null) {
                        tail = temp;
                    }
                    break;
                }
                temp = temp.next;
            }
        }
    }

    void removeAt(int index) {
        if (index == 0) {
            removeFirst();
        } else {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = temp.next.next;
            if (temp.next == null) {
                tail = temp;
            }
        }
    }
}
```

b. SLLMain.java

```
public class SLLMain {
    public static void main(String[] args) {
        SingleLinkedList singLL = new SingleLinkedList();
        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.insertAfter(700, 999);
        singLL.print();
        singLL.insertAt(3, 833);
        singLL.print();

        System.out.println("Data pada indeks ke-1=" +
singLL.getData(1));
        System.out.println("Data 3 berada pada indeks ke-" +
singLL.indexOf(760));
        singLL.remove(999);
        singLL.print();
        singLL.removeAt(0);
        singLL.print();
        singLL.removeFirst();
        singLL.print();
        singLL.removeLast();
        singLL.print();
    }
}
```

2.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
Linked list kosong
Isi Linked List:        890
Isi Linked List:        890     760
Isi Linked List:        700     890     760
Isi Linked List:        700     999     890     760
Isi Linked List:        700     999     890     833     760
Data pada indeks ke-1=999
Data 3 berada pada indeks ke-4
Isi Linked List:        700     890     833     760
Isi Linked List:        890     833     760
Isi Linked List:        833     760
Isi Linked List:        833
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
8b75
Linked List kosong
Isi Linked List: 890
Isi Linked List: 890    760
Isi Linked List: 700    890     760
Isi Linked List: 700    999     890     760
Isi Linked List: 700    999     890     833     760
Data pada indeks ke-1=999
Data 3 berada pada indeks ke-4
Isi Linked List: 700    890     833     760
Isi Linked List: 890    833     760
Isi Linked List: 833    760
Isi Linked List: 833
PS C:\Users\asus\Documents\Semester2\Algoritma dan St
```

2.2.3 Pertanyaan

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!
   → Dalam metode remove, keyword break digunakan untuk mencegah iterasi melalui linked list yang terhubung setelah operasi penghapusan selesai. Ini dilakukan untuk menghindari iterasi yang tidak perlu setelah elemen yang tepat dihapus.

2. Jelaskan kegunaan kode di bawah pada method remove

```
else if (temp.next.data == key) {
  temp.next = temp.next.next;
```

   → Kode ini termasuk dalam blok else if dan dijalankan jika menemukan node dengan nilai key yang sesuai, yang menunjukkan bahwa node yang ingin hapus adalah temp.next. Cara yang efektif dan efisien untuk menghapus node dari linked list terkait adalah dengan menggunakan kode temp.next = temp.next.next;. Ini memungkinkan menghapus node dari daftar terkait tanpa perlu alokasi memori tambahan atau mengganggu keterhubungan struktur daftar terkait.

3. Tugas Praktikum

1. Implementasikan ilustrasi Linked List Berikut. Gunakan 4 macam penambahan data yang telah dipelajari sebelumnya untuk menginputkan data.

a. dataNode

```
public class dataNode {
    String nama;
    String nim;
    dataNode next;

    dataNode(String nama, String nim) {
        this.nama = nama;
        this.nim = nim;
        this.next = null;
    }
}
```

b. LinkedList

```
public class LinkedList {

    dataNode head;

    boolean isEmpty() {
        return head!= null;
    }

     void print() {
        if (isEmpty()) {
            dataNode tmp = head;
            System.out.println();
            System.out.print("Isi Linked List : ");
            System.out.println();
            while (tmp != null) {
                System.out.println();
                System.out.println("Nama \t: " + tmp.nama);
                System.out.println("NIM \t: " + tmp.nim);
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println();
            System.out.println("Linked List kosong");
        }
```

```
    }

    void addFirst(String nama, String nim) {
        dataNode newDN = new dataNode(nama, nim);
        if (isEmpty()) {
            newDN.next = head;
            head = newDN;
        } else {
            head = newDN;
        }
    }

    void addLast(String nama, String nim) {
        dataNode newDN = new dataNode(nama, nim);
        if (head == null) {
            head = newDN;
            return;
        }
        dataNode last = head;
        while (last.next != null) {
            last = last.next;
        }
    }

    void insertAfter(dataNode prevNode, String nama, String
nim) {
        if (prevNode == null) {
            System.out.println("Node sebelumnya tidak boleh
null");
            return;
        }
        dataNode newNode = new dataNode(nama, nim);
        newNode.next = prevNode.next;
        prevNode.next = newNode;
    }

    void insertAt(int index, String nama, String nim) {
        if (index < 0) {
            System.out.println("Indeks tidak valid");
            return;
        }
        if (index == 0) {
            addFirst(nama, nim);
            return;
        }
        dataNode newNode = new dataNode(nama, nim);
        dataNode current = head;
        for (int i = 0; i < index - 1; i++) {
            if (current == null) {
                System.out.println("Indeks melebihi panjang
linked list");
                return;
            }
```

```
            current = current.next;
        }
        newNode.next = current.next;
        current.next = newNode;
    }

}
```

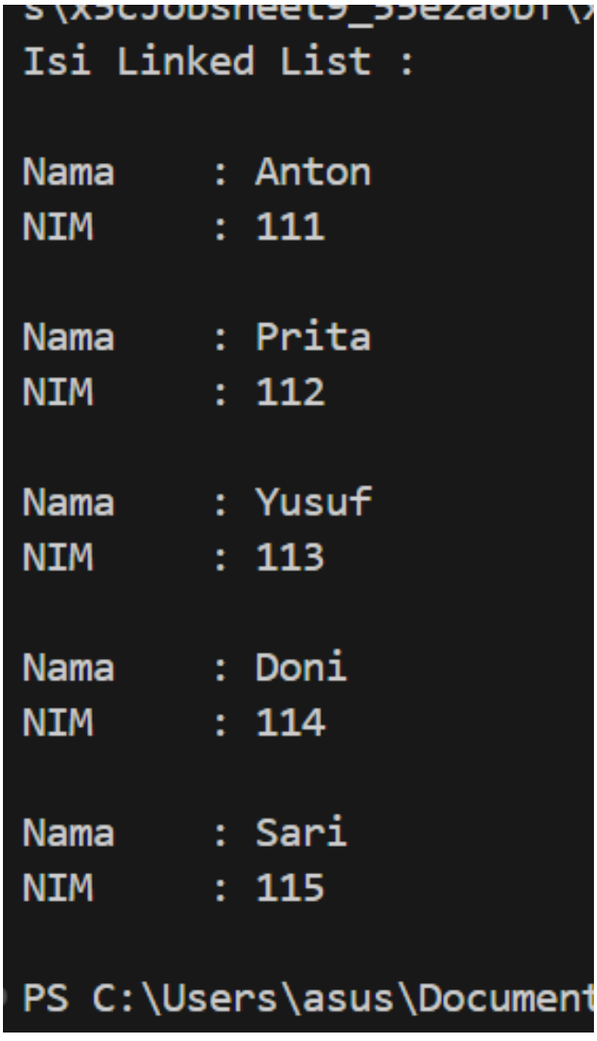
c. LLMain

```
public class LLMain {

    public static void main(String[] args) {
        LinkedList dataList = new LinkedList();

        dataList.addLast("Sari", "115");
        dataList.addFirst("Anton", "111");
        dataList.insertAfter(dataList.head, "Prita", "112");
        dataList.insertAt(2, "Yusuf", "113");
        dataList.insertAt(3, "Doni", "114");
        dataList.print();
    }
}
```

d. Hasil Praktikum



2. Buatlah implementasi program antrian layanan unit kemahasiswaan sesuai dengan kondisi yang ditunjukkan pada soal nomor 1! Ketentuan
    a. Implementasi antrian menggunakan Queue berbasis Linked List!
    b. Program merupakan proyek baru, bukan modifikasi dari soal nomor 1!

a. Node

```
package Praktikum2;

public class Node {

    String nama;
    String nim;
    Node next;

    Node(String nama, String nim) {
        this.nama = nama;
        this.nim = nim;
        this.next = null;
    }
}
```

b. Queue

```
package Praktikum2;

public class Queue {

    private Node front; //Node awal
    private Node rear; // Node Akhir

    Queue() {
        this.front = null;
        this.rear = null;
    }

    boolean isEmpty() {
        return front == null;
    }

    void displayQueue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        Node temp = front;
        System.out.println();
        System.out.println("Antrian Layanan Unit Kemahasiswaan :");
        while (temp != null) {
            System.out.println("Nama \t: " + temp.nama);
            System.out.println("NIM \t: " + temp.nim);
            System.out.println();
            temp = temp.next;
        }
```

```
    }

    void enqueue(String nama, String nim) {
        Node NQ = new Node(nama, nim);
        if (rear == null) {
            front = rear = NQ;
            return;
        }
        rear.next = NQ;
        rear = NQ;
    }

    void dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong, tidak ada yang bisa
dihapus.");
            return;
        }
        Node temp = front;
        front = front.next;
        if (front == null) {
            rear = null;
        }
        System.out.println("Nama \t: " + temp.nama);
        System.out.println("NIM \t: " + temp.nim);
        System.out.println();
    }

}
```

c. NQMain

```
package Praktikum2;

public class NQMain {

    public static void main(String[] args) {

        Queue QNList = new Queue();

        QNList.enqueue("Anton", "111");
        QNList.enqueue("Prita", "112");
        QNList.enqueue("Yusuf", "113");
        QNList.enqueue("Doni", "114");
        QNList.enqueue("Sari", "115");

        QNList.displayQueue();

        QNList.dequeue();

        QNList.displayQueue();
    }
```

```
}
```

d. Hasil Praktikum

```
Antrian Layanan Unit Kemahasiswaan :
Nama    : Anton
NIM     : 111

Nama    : Prita
NIM     : 112

Nama    : Yusuf
NIM     : 113

Nama    : Doni
NIM     : 114

Nama    : Sari
NIM     : 115

Nama    : Anton
NIM     : 111


Antrian Layanan Unit Kemahasiswaan :
Nama    : Prita
NIM     : 112

Nama    : Yusuf
NIM     : 113

Nama    : Doni
NIM     : 114

Nama    : Sari
NIM     : 115

PS C:\Users\asus\Documents\Semester2\Al
```