LAPORAN PRAKTIKUM ALGORITMA STUKTUR DATA

JOB SHEET 11
LINKED LISTS

Oleh:
SELA AULIA SISWANTO NIM. 1941720196

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
APRIL 2020

Praktikum

```
package percobaan;

/**
 *
 * @author Asus
 */
public class Node {

    int data;
    Node next;

    public Node( int data, Node next){
        this.data = data;
        this.next = next;
    }

}
```

```
package percobaan;

/**
 *
 * @author Asus
 */
public class LinkedLists {

    Node head;
    int size;

    public LinkedLists() {
        head = null;
```

```
    size = 0;
}

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

public void addFirst(int item) {
    head = new Node(item, head);
    size++;
}

public void add(int item, int index) throws Exception {
    if (index < 0 || index > size) {
        throw new Exception("Nilai index di luar batas");
    }
    if (isEmpty() || index == 0) {
        addFirst(item);
    } else {
        Node tmp = head;
        for (int i = 1; i < index; i++) {
            tmp = tmp.next;
        }
        Node next = (tmp == null) ? null : tmp.next;
        tmp.next = new Node(item, next);
    }
    size++;
}

public void addLast(int item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
```

```
            Node tmp = head;
            while (tmp.next != null) {
                tmp = tmp.next;
            }
            tmp.next = new Node(item, null);
            size++;
        }
    }

    public int getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("LinkedLists Kosong");
        }
        return head.data;
    }

    public int getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("LinkedLists kosong");
        }
        Node tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai index di luar batas");
        }
        Node tmp = head;
        for (int i = 0; i < index; i++) {
```

```
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public void remove(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai index di luar batas");
        }
        Node prev = head;
        Node cur = head.next;
        for (int i = 1; i < index; i++) {
            prev = cur;
            cur = prev.next;
        }
        prev.next = cur.next;
        size--;
    }

    public void removeFirst() throws Exception {
        head = head.next;
        size--;
    }

    public void clear() {
        head = null;
    }

    public void print() {
        if (!isEmpty()) {
            Node tmp = head;
            while (tmp != null) {
                System.out.println(tmp.data + "\t");
```

```
            tmp = tmp.next;
        }
        System.out.println();
    } else {
        System.out.println("LinkedLists kosong");
    }
  }
  }
```

```
package percobaan;
import java.util.Scanner;
/**
 *
 * @author Asus
 */
public class MainLinkedLists {
   public static void Menu() {

      System.out.println("====================================");
      System.out.println(" MENU ");
      System.out.println("1. Tambah");
      System.out.println("2. Hapus");
      System.out.println("3. Cari");
      System.out.println("4. Keluar");

      System.out.println("====================================");
   }

   public static void MenuAdd() {

      System.out.println("====================================");
      System.out.println(" MENU ");
      System.out.println("1. Tambah (First)");
```

```
        System.out.println("2. Tambah (Index)");
        System.out.println("3. Tambah (Last)");

        System.out.println("====================================");
    }

    public static void MenuHapus() {

        System.out.println("====================================");
        System.out.println(" MENU ");
        System.out.println("1. Hapus (Index)");
        System.out.println("2. Hapus (Key)");
        System.out.println("3.. Clear");

        System.out.println("====================================");
    }

    public static void MenuCari() {

        System.out.println("====================================");
        System.out.println(" MENU ");
        System.out.println("1. Cari (Index)");
        System.out.println("2. Cari (Key)");

        System.out.println("====================================");
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int pilih, sub;
        int dt, idx;
        LinkedLists data = new LinkedLists();
        try {
            do {
```

```
            Menu();
            System.out.println("Masukkan Pilihan Anda : ");
            pilih = sc.nextInt();
            switch (pilih) {
               case 1:
                  do {
                     MenuAdd();
                     System.out.println("Masukkan Pilihan Anda :");
                     sub = sc.nextInt();
                     switch (sub) {
                        case 1:
                           System.out.println("Masukkan Data : ");
                           dt = sc.nextInt();
                           System.out.println("======================");
                           data.addFirst(dt);
                           data.print();
                           break;
                        case 2:
                           System.out.println("Masukkan Data : ");
                           dt = sc.nextInt();
                           System.out.println("Masukkan Index : ");
                           idx = sc.nextInt();

                           System.out.println("========================");
                           data.add(dt, idx);
                           data.print();
                           break;
                        case 3:
                           System.out.println("Masukkan Data : ");
                           dt = sc.nextInt();
                           data.addFirst(dt);
                           data.print();
                           break;
```

```
                    case 4:
                        System.out.println("Masukkan Data : ");
                        dt = sc.nextInt();
                        data.addLast(dt);
                        data.print();
                        break;
                    }
                } while (sub != 0);
                break;
            case 3:
                do {
                    MenuCari();
                    System.out.println("Masukkan Pilihan Anda : ");
                    sub = sc.nextInt();
                    switch (sub) {
                        case 1:
                            System.out.println("Masukkan Index : ");
                            idx = sc.nextInt();
                            data.remove(idx);
                            data.print();
                            break;
                        case 2:
                            System.out.println("Masukkan Data : ");
                            dt = sc.nextInt();
                            data.removeFirst();
                            data.print();
                            break;
                    }
                } while (sub != 0);
                break;
        }
    } while (pilih != 0);
} catch (Exception e){
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

Pertanyaan

1. Mengapa pada proses traverse nilai head perlu disimpan terlebih dahulu dalam variabel tmp ?

   Jawab : Transver adalah operasi yang digunakan untuk mencetak dam memproses penambahan data di akhir linked lists. Jadi fungsi head perlu disimpan di variabel tmp adalah agar ketika penambahan data baru head tidak berubah. Data yang baru akan menempati tmp saja dan head tetap berisi data.
2. Apa kekurangan implementasi single LinkedLists tanpa penunjuk tail ?

   Jawab : Tidak akan bisa menambah data baru di belakang, karena perubahan data baru membutuhkan tail yang mengikat node baru. Secara singkatnya agar head tidak berubah atau bergeser.
3. Tambahkan implementasi method addByValue berdasarkan nilai yang dicari! Node baru akan ditambahkan setelah node yang dicari ditemukan.

```
package percobaan;


/**
 *
 * @author Asus
 */
public class LinkedLists {


    Node head;
    int size;


    public LinkedLists() {
        head = null;
        size = 0;
```

```
    }

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

    public void addFirst(int item) {
        head = new Node(item, head);
        size++;
    }

    public void add(int item, int index) throws Exception {
        if (index < 0 || index > size) {
            throw new Exception("Nilai index di luar batas");
        }
        if (isEmpty() || index == 0) {
            addFirst(item);
        } else {
            Node tmp = head;
            for (int i = 1; i < index; i++) {
                tmp = tmp.next;
            }
            Node next = (tmp == null) ? null : tmp.next;
            tmp.next = new Node(item, next);
        }
        size++;
    }

    public void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node tmp = head;
```

```
            while (tmp.next != null) {
                tmp = tmp.next;
            }
            tmp.next = new Node(item, null);
            size++;
        }
    }

    public int getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("LinkedLists Kosong");
        }
        return head.data;
    }

    public int getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("LinkedLists kosong");
        }
        Node tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai index di luar batas");
        }
        Node tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
```

```
    }
    return tmp.data;
}

public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai index di luar batas");
    }
    Node prev = head;
    Node cur = head.next;
    for (int i = 1; i < index; i++) {
        prev = cur;
        cur = prev.next;
    }
    prev.next = cur.next;
    size--;
}

public void removeFirst() throws Exception {
    head = head.next;
    size--;
}

public void clear() {
    head = null;
}

public void print() {
    if (!isEmpty()) {
        Node tmp = head;
        while (tmp != null) {
            System.out.println(tmp.data + "\t");
            tmp = tmp.next;
```

```
            }
            System.out.println();
        } else {
            System.out.println("LinkedLists kosong");
        }
    }

    public void addByValue(int cari, int item) throws Exception{
        boolean a = true;
        Node tmp = head;
        while (tmp.data != cari){
            tmp = tmp.next;
        }
        tmp.next = new Node (item, null);
        size++;
    }

    public void removeValue(int data) throws Exception{
        Node prev = head;
        Node cur = head.next;
        for(int i =1; i < size; i++){
            if (data != prev.data){
                prev = prev;
                cur = prev.next;
            }
        }
        prev.next = cur.next;
        size--;
    }

     public void cari(int index) throws Exception {
        if (index < 0 || index > size) {
            throw new Exception("Nilai index diluar batas");
```

```
        } else {
            Node tmp = head;
            for (int i = 1; i < index; i++) {
                tmp = tmp.next;
            }
            System.out.println("Data pada indeks ke-" + index + " adalah : " + tmp.data);
        }
    }

    public void cariKey(int cari) throws Exception {
        Node tmp = head;
        int index = 0;
        boolean ditemukan = false;
        if (isEmpty()) {
            System.out.println("LinkedLists kosong");
        } else {
            while (tmp.next != null) {
                tmp = tmp.next;
                index++;
                if (head.data == cari) {
                    ditemukan = true;
                    break;
                } else if (tmp.data == cari) {
                    ditemukan = true;
                    index++;
                    break;
                }
            }
        }
        if (ditemukan) {
            System.out.println("Data " + cari + " ditemukan pada indeks ke-" + index);
        } else {
            throw new Exception("Data tidak ditemukan!");
```

```
        }
    }

}
```

Tugas Praktikum

1. Buatlah program daftar mahasiswa menggunakan LinkedLists! Mahasiswa memiliki atribut NIM, nama, dan alamat tinggal.
   Jawab :

```
package Tugas1;

/**
 *
 * @author Asus
 */
public class Mahasiswa {

    String nim;
    String nama;

    Mahasiswa(String nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }
}
```

```
package Tugas1;

/**
 *
 * @author Asus
```

```
 */
public class NodeMahasiswa {

    Mahasiswa data;
    NodeMahasiswa next;

    public NodeMahasiswa( Mahasiswa data, NodeMahasiswa next){
        this.data = data;
        this.next = next;
    }
}
```

```
public class MainMahasiswa {

    static void menu() {
        System.out.println("====================================");
        System.out.println("1. Tambah");
        System.out.println("2. Hapus");
        System.out.println("3. Cari");
        System.out.println("4. Print");
        System.out.println("5. Keluar");
        System.out.println("+ +");
        System.out.print("Pilih menu sesuai keinginan Anda : ");
    }

    static void menuTambah() {
        System.out.println("====================================");
        System.out.println("1. Add First");
        System.out.println("2. Add Index");
        System.out.println("3. Add Key");
        System.out.println("4. Add Last");
        System.out.println("+ +");
        System.out.print("Pilih menu sesuai keinginan Anda : ");
```

```
    }

    static void menuHapus() {
        System.out.println("====================================");
        System.out.println("1. Hapus Index");
        System.out.println("2. Hapus Key");
        System.out.println("3. Hapus First");
        System.out.println("4. Clear");
        System.out.println("+ +");
        System.out.print("Pilih menu sesuai keinginan Anda : ");
    }

    static void menuCari() {
        System.out.println("====================================");
        System.out.println("1. Cari Index");
        System.out.println("2. Cari Key");
        System.out.println("3. Get First");
        System.out.println("4. Get Last");
        System.out.println("+ +");
        System.out.print("Pilih menu sesuai keinginan Anda : ");
    }

    public static void main(String[] args) {
        LinkedListsMahasiswa data = new LinkedListsMahasiswa();
        Scanner scan = new Scanner(System.in);
        Scanner scanStr = new Scanner(System.in);
        int pil1 = 0;
        int pil2 = 0;
        String nama = " ";
        String nim = " ";
        int nilaiIndex = 0;
        try {
            do {
```

```
            menu();
            pil1 = scan.nextInt();
            switch (pil1) {
               case 1:
                  menuTambah();
                  pil2 = scan.nextInt();
                  switch (pil2) {
                     case 1:
                        System.out.print("Masukkan NIM Anda :");
                        nim = scanStr.nextLine();
                        System.out.print("Masukkan nama Anda : ");
                        nama = scanStr.nextLine();
                        data.addFirst(nim, nama);
                        break;
                     case 2:
                        System.out.print("Masukkan NIM Anda: ");
                        nim = scanStr.nextLine();
                        System.out.print("Masukkan nama Anda: ");
                        nama = scanStr.nextLine();
                        System.out.print("Masukkan index : ");
                        nilaiIndex = scan.nextInt();
                        data.add(nim, nama, nilaiIndex);
                        break;
                     case 3:
                        System.out.print("Cari dan Masukkan nim Anda : ");
                        nim = scanStr.nextLine();
                        data.addKey(nim);
                        break;
                     case 4:
                        System.out.print("Masukkan NIM Anda: ");
                        nim = scanStr.nextLine();
                        System.out.print("Masukkan nama Anda: ");
                        nama = scanStr.nextLine();
```

```
                    data.addLast(nim, nama);
                    break;
                default:
                    System.out.println("Pilihan Anda tidak tersedia ");
            }
            break;
        case 2:
            menuHapus();
            pil2 = scan.nextInt();
            switch (pil2) {
                case 1:
                    System.out.print("Masukkan index : ");
                    nilaiIndex = scan.nextInt();
                    data.remove(nilaiIndex);
                    break;
                case 2:
                    System.out.print("Cari dan Dihapus NIM : ");
                    nim = scanStr.nextLine();
                    data.removeKey(nim);
                    break;
                case 3:
                    data.removeFirst();
                    break;
                case 4:
                    data.clear();
                    break;
                default:
                    System.out.println("Pilihan tidak tersedia ");
            }
            break;
        case 3:
            menuCari();
            pil2 = scan.nextInt();
```

```
                switch (pil2) {
                    case 1:
                        System.out.print("Masukkan index : ");
                        nilaiIndex = scan.nextInt();
                        data.get(nilaiIndex);
                        System.out.println("Data : ");
                        System.out.println("NIM : " + data.get(nilaiIndex).nim);
                        System.out.println("Nama : " + data.get(nilaiIndex).nama);
                        break;
                    case 2:
                        System.out.print("Cari NIM : ");
                        nim = scanStr.nextLine();
                        int hasil = data.getKey(nim);
                        if (hasil != -1) {
                            System.out.println("Data ditemukan di index ke-" + hasil);
                        } else {
                            System.out.println("Tidak ditemukan ");
                        }
                        break;
                    case 3:
                        System.out.println("Data : ");
                        System.out.println("NIM : " + data.getFirst().nim);
                        System.out.println("Nama : " + data.getFirst().nama);
                        break;
                    case 4:
                        System.out.println("Data : ");
                        System.out.println("NIM : " + data.getFirst().nim);
                        System.out.println("Nama : " + data.getFirst().nama);
                        break;
                    default:
                        System.out.println("Pilihan tidak tersedia ");
                }
                break;
```

```
                case 4:
                    data.print();
                    break;
                }
            } while (pil1 != 5);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
public class LinkedListsMahasiswa {

    NodeMahasiswa head;
    int size;
    Mahasiswa mhs;

    LinkedListsMahasiswa() {
        head = null;
        size = 0;
    }

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

    public void addFirst(String nim, String nama) {
        mhs = new Mahasiswa(nim, nama);
        head = new NodeMahasiswa(mhs, head);
        size++;
    }

    public void add(String nim, String nama, int index) throws Exception {
```

```
        mhs = new Mahasiswa(nim, nama);
        if (index < 0 || index > size) {
            throw new Exception("Nilai index di luar batas");
        }
        if (isEmpty() || index == 0) {
            addFirst(nim, nama);
        } else {
            NodeMahasiswa tmp = head;
            for (int i = 1; i < index; i++) {
                tmp = tmp.next;
            }
            NodeMahasiswa next = (tmp == null) ? null : tmp.next;
            tmp.next = new NodeMahasiswa(mhs, next);
            size++;
        }
    }

    public void addLast(String nim, String nama) {
        mhs = new Mahasiswa(nim, nama);
        if (isEmpty()) {
            addFirst(nim, nama);
        } else {
            NodeMahasiswa tmp = head;
            while (tmp.next != null) {
                tmp = tmp.next;
            }
            tmp.next = new NodeMahasiswa(mhs, null);
            size++;
        }
    }

    public Mahasiswa getFirst() throws Exception {
        if (isEmpty()) {
```

```
            throw new Exception("LinkedList kosong");
        }
        return head.data;
    }

    public Mahasiswa getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("LinkedList kosong");
        }
        NodeMahasiswa tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public Mahasiswa get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai index di luar batas");
        }
        NodeMahasiswa tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public void addKey(String nim) throws Exception {
        NodeMahasiswa tmp = head;
        String nama = "";
        for (int i = 0; i < size; i++) {
            if (tmp.data.nim.equalsIgnoreCase(nim)) {
                nama = tmp.data.nama;
```

```
            mhs = new Mahasiswa(nim, nama);
            add(nim, nama, i + 1);
            break;
        }
        tmp = tmp.next;
    }
}

public int getKey(String nim) throws Exception {
    NodeMahasiswa tmp = head;
    int simpan = -1;
    for (int i = 0; i < size; i++) {
        if (tmp.data.nim.equalsIgnoreCase(nim)) {
            simpan = i;
            break;
        }
        tmp = tmp.next;
    }
    return simpan;
}

public void removeKey(String nim) throws Exception {
    NodeMahasiswa tmp = head;
    for (int i = 0; i < size; i++) {
        if (tmp.data.nim.equalsIgnoreCase(nim)) {
            remove(i);
            break;
        }
        tmp = tmp.next;
    }
}

public void remove(int index) throws Exception {
```

```
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai index di luar batas");
    }
    if (isEmpty() || index == 0) {
        removeFirst();
    } else {
        NodeMahasiswa prev = head;
        NodeMahasiswa cur = head.next;
        for (int i = 1; i < index; i++) {
            prev = cur;
            cur = prev.next;
        }
        prev.next = cur.next;
        size--;
    }
}

public void removeFirst() throws Exception {
    Mahasiswa tmp = getFirst();
    head = head.next;
    size--;
}

public void clear() {
    head = null;
    size = 0;
}

public void print() {
    if (!isEmpty()) {
        NodeMahasiswa tmp = head;
        while (tmp != null) {
            System.out.println("Nim : " + tmp.data.nim);
```

```
            System.out.println("Nama : " + tmp.data.nama);
            tmp = tmp.next;
        }
        System.out.println("");
    } else {
        System.out.println("LinkedList kosong");
    }
  }
}
```

2. Carilah studi kasus lain yang memanfaatkan Stack atau Queue (pilih salah satu) lalu buat program menggunakan konsep LinkedLists!

Jawab :

```
package Tugas2;

/**
 *
 * @author Asus
 */
public class NodeStack {

    public int data;
    public NodeStack next;
    public NodeStack prev;

    public NodeStack(int id) {
        this.data = id;
    }

    public void tampil() {
        System.out.println("{" + data + "}");
    }
```

```
}
```

```
package Tugas2;

/**
 *
 * @author Asus
 */
public class LinkedLists {

    private NodeStack top;
    private NodeStack bottom;

    public LinkedLists() {
        top = bottom = null;
    }

    public boolean isEmpty() {
        return (top == null);
    }

    public void push(int id) {
        NodeStack st = new NodeStack(id);
        if (top == null) {
            top = bottom = st;
        } else {
            top.next = st;
            st.prev = top;
            top = st;
        }
    }

    public NodeStack pop() {
```

```
        NodeStack tmp = null;
        if (top == null) {
            System.out.println("Stack Kosong");
        } else if (top == bottom) {
            tmp = top;
            top = bottom = null;
        } else {
            tmp = top;
            top = top.prev;
            top.next = null;
        }
        return tmp;
    }

    public void print() {
        NodeStack current = bottom;
        while (current != null) {
            current.tampil();
            current = current.next;
        }
        System.out.println("");

    }
}
```

```
public class MainStack {

    public static void main(String[] args) {
        LinkedLists data = new LinkedLists();

System.out.println("=============================================");
        System.out.println("     Inisialisasi Stack & Linked Lists         ");
```

```
System.out.println("==========================================");
        System.out.println("Data dimasukkan dalam source code");
        data.push(80);
        data.push(45);
        data.push(21);
        System.out.println("Tampilkan Stack");
        data.print();
        System.out.println("Hapus Stack dari Top");
        while (!data.isEmpty()) {
            NodeStack ns = data.pop();
            System.out.print("Hapus");
            ns.tampil();
            System.out.println("");
        }
        data.print();
    }
```