**LAPORAN PRAKTIKUM STRUKTUR DATA**

Modul ke : 8

Judul Praktikum : Double Linked List

Hari dan Tanggal Pelaksanaan : Kamis, 28 April 2022

Tempat Pelaksanaan : Lab Desain

Dosen Pengampu Praktikum : Khoirul Umam, S.Pd, M.Kom

Nama Mahasiswa Pelaksana : Andri Wijaksono

NIM Pelaksana : 362155401206

1. **Tugas Pendahuluan**

Tuliskan hasil pengerjaan Tugas Pendahuluan pada bagian ini.

Buatlah resume mengenai double linked list.

Pada awal mulanya linked list dibangun dari sekumpulan node dimana tiap node hanya mengandung satu buah pointer yang menunjuk ke satu node lainnya. Kumpulan node-node tersebut akhirnya membentuk sebuah model yang disebut sebagai single linked list. Akan tetapi model single linked list memiliki sebuah kelemahan, yaitu untuk menjelajahi tiap node hanya bisa dilakukan dalam satu arah yang dimulai dari node paling awal.

Untuk mengatasi kekurangan pada model tersebut maka dikembangkan kembali model linked list dengan dua buah pointer pada tiap node. Masing-masing pointer akan menunjuk ke node sebelumnya (previous) dan node setelahnya (next) seperti yang ditunjukkan oleh gambar di bawah ini. Seperti pada model single linked list, model double linked list juga dapat dibedakan menjadi double linked list non-circular (DLLNC) dan double linked list circular (DLLC).

* **Java Linked List**

Pada bahasa pemrograman Java telah tersedia built-in class LinkedList yang berada di dalam package java.util. Built-in class ini telah menyediakan beberapa method untuk menambahkan/menyisipkan node baru, menghapus node, hingga mencari bahkan mengurutkan node berdasarkan data yang disimpan. Class ini mengimplementasikan model double linked list.

1. **Kegiatan Praktikum**

Cantumkan apa saja yang dilakukan pada latihan-latihan praktikum, *source code* yang dipakai, *screen shot* hasil eksekusi kode, dan jawaban dari pertanyaan-pertanyaan yang muncul pada ti­ap kegiatan latihan.

**Latihan 1: Membuat double linked list non-circular (DLLNC)**

1. Buat sebuah file bernama **DNode**.**java** di dalam folder yang sama dengan file-file modul praktikum **Single** **Linked** **List** sebelumnya kemudian tuliskan kode berikut:

public class DNode<T> {

private T value;

private DNode<T> next;

private DNode<T> prev;

public DNode(T value) {

this.value = value;

}

public T getValue() {

return value;

}

public void setNext(DNode<T> nextNode) {

next = nextNode;

}

public DNode<T> getNext() {

return next;

}

public boolean hasNext() {

return next != null;

}

public void setPrev(DNode<T> prevNode) {

prev = prevNode;

}

public DNode<T> getPrev() {

return prev;

}

public boolean hasPrev() {

return prev != null;

}

}

2. Masih di dalam folder yang sama, buatlah sebuah file baru untuk mendeklarasikan class double linked list dengan nama **DoubleLinkedListNonCircular**.**java** dan tuliskan kode berikut ke dalamnya:

public class DoubleLinkedListNonCircular<T> implements LinkedList<T> {

private DNode<T> head = null;

private DNode<T> tail = null;

public void addFirst(T value) {

DNode<T> node = new DNode<T>(value);

if (head == null) {

head = node;

tail = node;

} else {

node.setNext(head);

head.setPrev(node);

head = node;

}

}

public void addLast(T value) {

DNode<T> node = new DNode<T>(value);

if (tail == null) {

head = node;

tail = node;

} else {

tail.setNext(node);

node.setPrev(tail);

tail = node;

}

}

public T removeFirst() {

if (head == null)

return null;

else {

DNode<T> removed = head;

if (head == tail) {

head = null;

tail = null;

} else {

head = head.getNext();

head.setPrev(null);

removed.setNext(null);

}

return removed.getValue();

}

}

public T removeLast() {

if (tail == null)

return null;

else {

DNode<T> removed = tail;

if (head == tail) {

head = null;

tail = null;

} else {

tail = tail.getPrev();

tail.setNext(null);

removed.setPrev(null);

}

return removed.getValue();

}

}

public boolean contains(T data) {

boolean found = false;

DNode<T> search = head;

while (search != null && !found) {

found = search.getValue().equals(data);

search = search.getNext();

}

return found;

}

public String toString() {

String str = "";

DNode<T> reader = head;

while (reader != null) {

str += reader.getValue();

if (reader.hasNext())

str += ", ";

reader = reader.getNext();

}

return str;

}

public T get(int position) {

return null;

}

public void add(int position, T value) {

}

public T remove(int position) {

return null;

}

}

public class TesLinkedList {

    public static void main(String[] args) {

        LinkedList<Character> list = new

        DoubleLinkedListNonCircular<Character>();

    System.out.println("Add first : A");

    list.addFirst('A');

    System.out.println("Add last : B");

    list.addLast('B');

    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Add first : C");

    list.addFirst('C');

    System.out.println("Add first : D");

    list.addFirst('D');

    System.out.println("Add last : E");

    list.addLast('E');

    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Remove first : " + list.removeFirst() + "removed");

    System.out.println("Remove last : " + list.removeLast() + " removed");

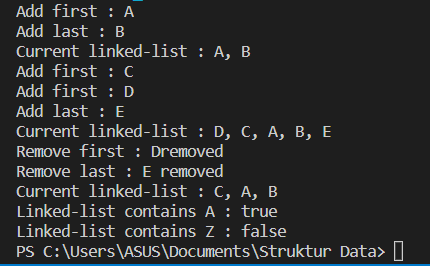
    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Linked-list contains A : " + list.contains('A'));

    System.out.println("Linked-list contains Z : " + list.contains('Z'));

    }

}



5. Perhatikan hasilnya, apakah ada perbedaan susunan data yang disimpan ke dalam double linked list jika dibandingkan dengan single linked list pada modul sebelumnya?

Jawab: Susunan data tidak berbeda dengan single linked list

6. Apa peran properti next dan prev pada class DNode?

Jawab: Peran next digunakan untuk mengarahkan node ke perintah selanjutnya, sedangkan prev digunakan untuk mengembalikan node ke posisi awal.

**Latihan 2: Membuat double linked list circular (DLLC)**

1. Buat file baru bernama DoubleLinkedListCircular.java dan tuliskan kode berikut:

public class DoubleLinkedListCircular<T> implements LinkedList<T> {

private DNode<T> head = null;

private DNode<T> tail = null;

public void addFirst(T value) {

DNode<T> node = new DNode<T>(value);

if (head == null) {

node.setNext(node);

node.setPrev(node);

head = node;

tail = node;

} else {

node.setNext(head);

node.setPrev(tail);

head.setPrev(node);

tail.setNext(node);

head = node;

}

}

public void addLast(T value) {

DNode<T> node = new DNode<T>(value);

if (tail == null) {

node.setNext(node);

node.setPrev(node);

head = node;

tail = node;

} else {

node.setNext(head);

node.setPrev(tail);

tail.setNext(node);

head.setPrev(node);

tail = node;

}

}

public T removeFirst() {

if (head == null)

return null;

else {

DNode<T> removed = head;

if (head == tail) {

head = null;

tail = null;

} else {

head = head.getNext();

head.setPrev(tail);

tail.setNext(head);

}

removed.setNext(removed);

removed.setPrev(removed);

return removed.getValue();

}

}

public T removeLast() {

if (tail == null)

return null;

else {

DNode<T> removed = tail;

if (head == tail) {

head = null;

tail = null;

} else {

tail = tail.getPrev();

tail.setNext(head);

head.setPrev(tail);

}

removed.setNext(removed);

removed.setPrev(removed);

return removed.getValue();

}

}

public boolean contains(T data) {

boolean found = false;

DNode<T> search = head;

if (search != null) {

do {

found = search.getValue().equals(data);

search = search.getNext();

} while (search != head && !found);

}

return found;

}

public String toString() {

String str = "";

DNode<T> reader = head;

if (reader != null) {

do {

str += reader.getValue();

if (reader.hasNext())

str += ", ";

reader = reader.getNext();

} while (reader != tail);

}

return str;

}

public T get(int position) {

return null;

}

public void add(int position, T value) {

}

public T remove(int position) {

return null;

}

}

public class TesLinkedList {

    public static void main(String[] args) {

        LinkedList<Character> list = new

        DoubleLinkedListCircular<Character>();

    System.out.println("Add first : A");

    list.addFirst('A');

    System.out.println("Add last : B");

    list.addLast('B');

    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Add first : C");

    list.addFirst('C');

    System.out.println("Add first : D");

    list.addFirst('D');

    System.out.println("Add last : E");

    list.addLast('E');

    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Remove first : " + list.removeFirst() + "removed");

    System.out.println("Remove last : " + list.removeLast() + " removed");

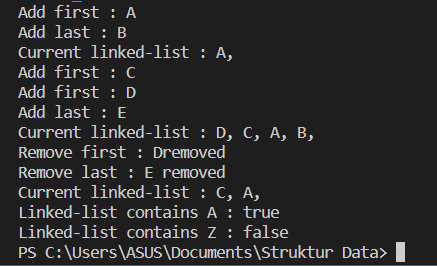
    System.out.println("Current linked-list : " + list.toString());

    System.out.println("Linked-list contains A : " + list.contains('A'));

    System.out.println("Linked-list contains Z : " + list.contains('Z'));

    }

}



4. Apa yang membedakan **DLLC** ini jika dibandingkan dengan **DLLNC** pada latihan sebelumnya?

Jawab: Pada latihan sebelumnya. Current linked list ada C, A, B. pada latihan ini hanya ada C, A

**Latihan 3: Menggunakan Java Linked List**

1. Buat sebuah file bernama **JavaLinkedList**.**java** kemudian tuliskan kode berikut:

import java.util.LinkedList;

public class JavaLinkedList {

public static void main(String[] args) {

LinkedList<Character> list = new LinkedList<Character>();

System.out.println("Add first : A");

list.addFirst('A');

System.out.println("Add last : B");

list.addLast('B');

System.out.println("Current linked-list : " + list.toString());

System.out.println("Add first : C");

list.addFirst('C');

System.out.println("Add first : D");

list.addFirst('D');

System.out.println("Add last : E");

list.addLast('E');

System.out.println("Current linked-list : " + list.toString());

System.out.println("Remove first : " + list.removeFirst() + "removed");

System.out.println("Remove last : " + list.removeLast() + " removed");

System.out.println("Current linked-list : " + list.toString());

System.out.println("Object value on 0 : " + list.get(0));

System.out.println("Object value on 1 : " + list.get(1));

System.out.println("Object value on 2 : " + list.get(2));

System.out.println("Remove object on 1 : " + list.remove(1) + "removed");

System.out.println("Current linked-list : " + list.toString());

System.out.println("Add object on 1 : Z");

list.add(1, 'Z');

System.out.println("Add object on 3 : X");

list.add(3, 'X');

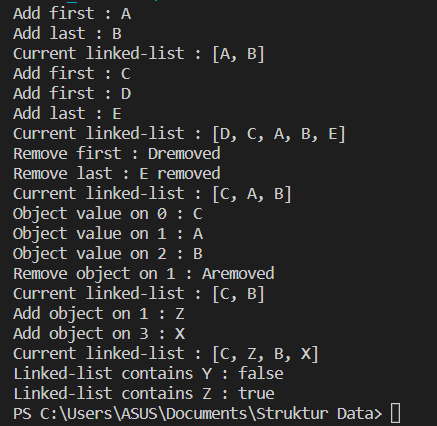
System.out.println("Current linked-list : " + list.toString());

System.out.println("Linked-list contains Y : " + list.contains('Y'));

System.out.println("Linked-list contains Z : " + list.contains('Z'));

}

}

****

3. Perhatikan hasilnya, apakah terdapat perbedaan dengan single linked list maupun double linked list yang telah dibuat sebelumnya?

Jawab: Pada Single Linked List hasil yang ditampilkan

4. Apa perbedaan antara method add dengan addFirst maupun addLast?

Jawab: Method add digunakan untuk menambahkan node baru

5. Apa perbedaan antara method remove dengan removeFirst maupun removeLast?

Jawab: Method remove digunakan untuk menghapus node

1. **Tugas Praktikum**

Tuliskan dan jabarkan hasil pengerjaan Tugas Praktikum yang tertera di dalam modul lengkap dengan *source code* yang digunakan.

Lengkapi deklarasi method **get**, **add**, dan **remove** pada latihan DLLNC maupun DLLC di atas agar dapat digunakan untuk mendapatkan data pada node dengan posisi tertentu, memasukkan data pada node dengan posisi tertentu, dan juga menghapus node data pada posisi tertentu sehingga dapat diperoleh hasil yang sama seperti jika langsung memanfaatkan **Java** **Linked** **List**.