LAPORAN PRAKTIKUM STRUKTUR DATA

Modul ke	: 7
Judul Praktikum	: Single Linked List
Hari dan Tanggal Pelaksanaan	: Rabu, 27 April 2022
Tempat Pelaksanaan	: Lab Desain
Dosen Pengampu Praktikum	: Khoirul Umam, S.Pd, M.Kom
Nama Mahasiswa Pelaksana	: Andri Wijaksono
NIM Pelaksana	: 362155401206

A. Tugas Pendahuluan

Tuliskan hasil pengerjaan Tugas Pendahuluan pada bagian ini.

Buatlah resume mengenai single linked list.

Single linked list terbentuk ketika satu objek saling bertaut dengan objek lainnya menggunakan satu tautan (link) pada tiap objeknya. Untuk dapat mengakses setiap node hanya dapat dapat dilakukan node paling awal.

Single linked list tediri dari dua macam. Yaitu single linked list circular dan single linked list non circular.

- Single linked list circular. Ketika setiap node bertautan dengan node lainnya, maka untuk node paling akhir akan betaut pada null atau tidak bertautan dengan node lainnya. Aksesnya akan berakhir di node terakhir itu juga.

Single linked list circular. Node terakhir akan bertautan dengan node paling awal. Konsepnya seperti perulangan atau memutar.

B. 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 Single Linked List Non-Circular (SLLNC)

1. Deklarasikan class **SNode** yang akan digunakan untuk membentuk objek-objek node pada SLLNC dengan menuliskan kode berikut ke dalam file **SNode.java**

```
public class SNode<T> {
private T value;
private SNode<T> next;
public SNode(T value) {
this.value = value;
}
public T getValue() {
return value;
}
public void setNext(SNode<T> nextNode) {
```

```
next = nextNode;
}
public SNode<T> getNext() {
return next;
}
public boolean hasNext() {
return next != null;
}
}
```

2. Deklarasikan interface **LinkedList** pada file **LinkedList.java** dengan menuliskan kode berikut:

```
public interface LinkedList<T> {
    public void addFirst(T value);
    public void addLast(T value);
    public T removeFirst();
    public T removeLast();
    public boolean contains(T data);
    public String toString();
}
```

3. Buat file bernama **SingleLinkedListNonCircular.java** kemudian tuliskan kode berikut ke dalamnya:

```
public class SingleLinkedListNonCircular<T> implements LinkedList<T> {
    private SNode<T> head = null;
    private SNode<T> tail = null;

    public void addFirst(T value) {
        SNode<T> node = new SNode<T>(value);

        if (head == null) {
            head = node;
            tail = node;
        } else {
            node.setNext(head);
            head = node;
        }
    }
    public void addLast(T value) {
        SNode<T> node = new SNode<T>(value);

        if (head == null) {
            head = node;
            tail = node;
        } else {
```

```
            tail.setNext(node);
            tail = node;
        }
    }
    public T removeFirst() {
        if (head == null)
            return null;
        else {
            SNode<T> removed = head;

        if (head == tail) {
            head = null;
            tail = null;
        } else {
            head = head.getNext();
            removed.setNext(null);
        }
        return removed.getValue();
    }
}

public T removeLast() {
    if (tail == null)
        return null;
    else {
        SNode<T> removed = tail;

        if (head == tail) {
            head = null;
            tail = null;
        } else {
            SNode<T> temp = head;

            while (temp.getNext() != tail)
                temp = temp.getNext();

            tail = temp;
            tail.setNext(null);
        }
        return removed.getValue();
    }
}
public boolean contains(T data) {
    boolean found = false;
    SNode<T> search = head;

    while (search != null && !found) {
        found = search.getValue().equals(data);
```

```
        search = search.getNext();
    }

    return found;
}

public String toString() {
    String str = "";
    SNode<T> reader = head;

    while (reader != null) {
        str += (reader.getValue() + "->");
        reader = reader.getNext();
    }
    return str;
    }
}
```

4. Siapkan testing code di dalam file **TesLinkedList.java** sebagai berikut:

```
public class TesLinkedList {
    public static void main(String[] args) {
        LinkedList<Character> list = new
SingleLinkedListNonCircular<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'));
    }
}
```

```
Add first : A
Add last : B
Current linked-list : A->B->
Add first : C
Add first : D
Add last : E
Current linked-list : D->C->A->B->E->
Remove first : Dremoved
Remove last : E removed
Current linked-list : C->A->B->
Linked-list contains A : true
Linked-list contains Z : false
PS C:\Users\ASUS\Documents\Struktur Data\modul7>
```

6. Mengapa tipe data dari properti **next** pada class **SNode** adalah SNode itu sendiri?
Jawab: Menggunakan self-referrential classes dimana suatu kelas mempunyai field yang merefer ke dirinya sendiri.
7. Jelaskan kegunaan variabel **head** dan **tail** pada single linked list non-circular di atas!
Jawab: Head pada single linked list non-circular digunakan untuk menentukan node paling awal dan tail digunakan untuk menentukan node paling akhir.
8. Kemana pointer **next** pada **tail** mengarah untuk kasus ini?
Jawab: Pointer next pada tail akan mengarah ke arah null dan tidak akan mengarah ke head atau node paling awal.
9. Apakah ada batasan jumlah objek/data yang dapat dimasukkan ke dalam linked list tersebut? Jelaskan!
Jawab: Tidak ada batasan jumlah objek / data. Karena di dalam linked list tidak dideklarasikan jumlah objek.

Latihan 2: Membuat Single Linked List Circular (SLLC)

1. Di folder yang sama dengan class **SingleLinkedListNonCircular** sebelumnya, buatlah file baru bernama **SingleLinkedListCircular.java** dan tuliskan kode berikut ke dalamnya:

```java
public class SingleLinkedListCircular<T> implements LinkedList<T> {
    private SNode<T> head = null;
    private SNode<T> tail = null;
    public void addFirst(T value) {
    SNode<T> node = new SNode<T>(value);
    if (head == null) {
    head = node;
    tail = node;
    } else {
    node.setNext(head);
    head = node;
    }
    tail.setNext(head);
    }
    public void addLast(T value) {
    SNode<T> node = new SNode<T>(value);
    if (tail == null) {
    head = node;
```

```
    tail = node;
    } else {
    tail.setNext(node);
    tail = node;
    }
    tail.setNext(head);
    }
    public T removeFirst() {
    if (head == null)
    return null;
    else {
    SNode<T> removed = head;
    if (head == tail) {
    head = null;
    tail = null;
    } else {
    head = head.getNext();
    tail.setNext(head);
}
removed.setNext(removed);
return removed.getValue();
}
}
public T removeLast() {
if (tail == null)
return null;
else {
SNode<T> removed = tail;
if (head == tail) {
head = null;
tail = null;
} else {
SNode<T> temp = head;
while (temp.getNext() != tail)
temp = temp.getNext();
tail = temp;
tail.setNext(head);
}
removed.setNext(removed);
return removed.getValue();
}
}
public boolean contains(T data) {
boolean found = false;
SNode<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 = "";
SNode<T> reader = head;
if (reader != null) {
do {
str += (reader.getValue() + "->");
reader = reader.getNext();
} while (reader != head);
}
return str;
}
}
```

2. Modifikasi testing code pada file **TesLinkedList.java** agar linked list yang digunakan adalah SLLC seperti berikut:

```
public class TesLinkedList {
    public static void main(String[] args) {
        LinkedList<Character> list = new
SingleLinkedListCircular<Character>();
```

```
public class TesLinkedList {
    public static void main(String[] args) {
        LinkedList<Character> list = new
SingleLinkedListCircular<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'));
    }
}
```

```
Add first : A
Add last : B
Current linked-list : A->B->
Add first : C
Add first : D
Add last : E
Current linked-list : D->C->A->B->E->
Remove first : Dremoved
Remove last : E removed
Current linked-list : C->A->B->
Linked-list contains A : true
Linked-list contains Z : false
PS C:\Users\ASUS\Documents\Struktur Data>
```

4. Apakah nampak ada perbedaan hasil antara latihan SLLC ini dibandingkan dengan latihan SLLNC sebelumnya?
Jawab: Tidak. Karena model program yang dibuat sama. Hanya saja, konsepnya berbeda. Jika SLLC nextnya akan mengarah ke null. Sedangkan SLLNC nextnya akan mengarah ke node awal.
5. Kemana pointer **next** pada **tail** mengarah untuk kasus ini?
Jawab: Pointer next pada tail akan mengarah ke node awal yaitu D.

C. Tugas Praktikum

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

Modifikasi interface **LinkedList** dengan menambahakan method **get**, **add**, dan **remove** menjadi seperti berikut:

```
public interface LinkedList<T> {
public T get(int position);
public void add(int position, T value);
public T remove(int position);
public void addFirst(T value);
public void addLast(T value);
public T removeFirst();
public T removeLast();
public boolean contains(T data);
public String toString();
}
```

dimana method-method tambahan tersebut masing-masing menerima satu buah parameter integer yang menunjukkan suatu indeks posisi. Kemudian tambahkan deklarasi method-method tersebut di dalam class **SLLNC** maupun **SLLC** sehingga linked list tersebut dapat melakukan pembacaan data pada node di posisi tertentu, menyisipkan node pada posisi tertentu, dan menghapus node pada posisi tertentu. Penambahan dan penghapusan node pada posisi tertentu tersebut harus tetap dapat menjaga ketertautan tiap objek di dalam linked list.