PRAKTIKUM 15
SINGLE LINKED LIST 1

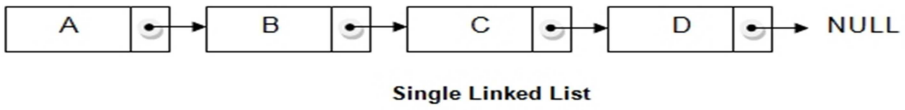
A. TUJUAN PEMBELAJARAN

Mahasiswa diharapkan mampu :

1. Memahami konsep SingleLinkedList untuk menambahkan node baru di awal, diakhir dan di index tertentu dan mengimplementasikan.
2. Memahami konsep SingleLinkedList untuk mendapatkan value pada node pada index tertentu pada SingleLinkedList dan mengimplementasikan.
3. Memahami konsep SingleLinkedList untuk merubah value pada node pada index tertentu pada SingleLinkedList dan mengimplementasikan.

B. DASAR TEORI

Linked list adalah sekumpulan elemen bertipe sama, yang mempunyai keterurutan tertentu, yang setiap elemennya terdiri dari dua bagian. Struktur berupa rangkaian elemen saling berkait dimana setiap elemen dihubungkan elemen lain melalui pointer. Pointer adalah alamat elemen.Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik walau tidak bersebelahan secara fisik di memori.



Gambar 1. Single Linked List

Terdapat tempat yang disediakan pada satu area memori tertentu untuk menyimpan data dikenal dengan sebutan node atau simpul.Setiap node memiliki pointer yang menunjuk ke simpul berikutnya sehingga terbentuk satu untaian, dengan demikian hanya diperlukan sebuah variabel pointer.Susunan berupa untaian semacam ini disebut Single Linked List (NULL memilik nilai khusus yang artinya tidak menunjuk ke mana-mana. Biasanya Linked List pada titik akhirnya akan menunjuk ke NULL). Salah satu kelemahan single linked list adalah pointer (penunjuk) hanya dapat bergerak satu arah saja, maju/mundur, atau kanan/kiri sehingga pencarian data pada single linked list hanya dapat bergerak dalam satu arah saja. Untuk mengatasi kelemahan tersebut, dapat menggunakan metode double linked list. Linked list ini dikenal dengan nama Linked list berpointer Ganda atau Double Linked List.

C. TUGAS PENDAHULUAN

1. Dengan menggunakan gambar, jelaskan langkah-langkah menambahkan node baru di depan SingleLinkedList.
2. Dengan menggunakan gambar, jelaskan langkah-langkah menambahkan node baru di depan SingleLinkedList.

D. PERCOBAAN

Pada percobaan SingleLinkedList (SLL), membuat class SingleLinkedList yang mengimplementasikan interface List<T>. Implementasikan method-method pada interface List<T> seperti tabel 1. Method yang bercetak tebal adalah method yang dibuat sendiri.
Pada praktikum ini yang dikerjakan adalah praktikum SLL 1.

Tabel 1. Method-method pada class SingleLinkedList<T>

`class SingleLinkedList<T> implements List<T>`		`Praktikum`
`private Node<T> front = null;`	Variable reference untuk menandai node awal dari SSL	**SLL 1**
`private int size;`	Variablel untuk mengetahui jumlah node pada SSL	
`private void addFirst(T item) {}`	Method addFirst() untuk menambahkan node baru di awal SSL	
`private void addLast(T item) {}`	Method addLast()untuk menambahkan node baru di akhir SSL	
`public boolean add(T e){}`	Method add()untuk menambahkan node baru di akhir SSL	
`public int size() {}`	Method size() untuk mengetahui jumlah node pada SSL	
`public void add(int index, T element) {}`	Method add(index,elemen) untuk menambahkan node baru dengan value elemen dengan pada index tertentu	
`public boolean isEmpty(){}`	Method isEmpty() untuk mengetahui apakah SSL masih null/kosong, jika ya maka mengembalikan nilai true, jika tidak null maka mengembalikan nilai false.	
`public T get(int`	Method get(index) untuk mendapatkan value	

`index) {}`	pada node pada index tertentu dari SSL	
`public T set(int index, T element){}`	Method set(index, element) untuk merubah value pada node pada index tertentu dengan element. Method ini mengembalikan value yang lama.	
`public String toString() {}`	Method toString() untuk mengubah objek SSL menjadi String	
`public Object[] toArray() {}`	Method toArray() untuk mengubah objek SSL menjadi array dengan tipe Object	**SLL 2**
`public boolean contains(Object o) {}`	Method contains() untuk mengecek apakah terdapat node o pada SSL	
`public boolean remove(Object o){}`	Method remove(Object o) untuk menghapus node o pada SSL	
`private T removeFirst() {}`	Method removeFirst() untuk menghapus node yang paling depan	
`private T removeLast() {}`	Method removeLast() untuk menghapus node yang paling akhir	
`public T remove(int index){}`	Method remove(index) untuk menghapus node pada index tertentu pada SSL	
`public int indexOf(Object o){}`	Method indexOf(Object o) untuk mendapatkan index pertama kali node o pada SSL	
`public int lastIndexOf(Object o){}`	Method lastIndexOf(Object o) untuk mendapatkan index terakhir node o pada SSL	
`public Iterator<T> iterator() {}`	Method iterator() untuk melakukan iterasi pada SSL	**SLL 3**

Percobaan 1. Membuat method addFirst() untuk menambahkan node baru di awal SLL

```
public class SingleLinkedList<T> implements List<T>{

    private Node<T> front = null;
    private int size;
```

```
    //untuk menambahkan node di depan
    private void addFirst(T item) {
        Node<T> newNode = new Node<T>(item);
        if (front == null) {
            front = newNode;
        } else {
            newNode.next = front;
            front = newNode;
        }
        size++;
    }
}
```

Percobaan 2. Mebuat method addLast()untuk menambahkan node baru di akhir SLL

```
public class SingleLinkedList<T> implements List<T>{

    //untuk menambahkan node di akhir
    private void addLast(T item) {
        Node<T> newNode = new Node<T>(item);
        if (front == null) {
            front = newNode;
        } else {
            Node<T> curr = front;
            while (curr.next != null) {
                curr = curr.next;
            }
            curr.next = newNode;
        }
        size++;
    }
}
```

Percobaan 3. Membuat method get(index) untuk mendapatkan value pada node pada index tertentu dari SLL

```
public class SingleLinkedList<T> implements List<T>{

public T get(int index) {
        Node<T> curr = front;
        T temp = null;
        int n = 0;
        if (index >= size) {
            return null;
        }
        while (curr != null) {
            if (n == index) {
                temp = curr.nodeValue;
                return temp;
            }
            curr = curr.next;
            n++;
        }
```

```
        return temp;
    }
}
```

Percobaan 4. Membuat method toString() untuk mengubah objek SLL menjadi String

```
public class SingleLinkedList<T> implements List<T>{


    @Override
    public String toString() {
        Node<T> curr = front;
        String str = "[" + curr.nodeValue;
        while (curr.next != null) {
            curr = curr.next;
            str += ", " + curr.nodeValue;
        }
        str += "]";
        return str;
    }
}
```

E. LATIHAN

Selesaikan method-method yang belum diimplementasikan pada Class SingleLinkedList

`class SingleLinkedList<T> implements List<T>`	
`public boolean add(T e){}`	Method add()untuk menambahkan node baru di akhir SSL
`public int size() {}`	Method size() untuk mengetahui jumlah node pada SSL
`public void add(int index, T element) {}`	Method add(index,elemen) untuk menambahkan node baru dengan value elemen dengan pada index tertentu
`public boolean isEmpty(){}`	Method isEmpty() untuk mengetahui apakah SSL masih null/kosong, jika ya maka mengembalikan nilai true, jika tidak null maka mengembalikan nilai false.
`public T set(int index, T element){}`	Method set(index, element) untuk merubah value pada node pada index tertentu dengan element. Method ini mengembalikan value yang lama.

Selanjutnya ujilah method-method pada Class SingleLinkedList dengan membuat class Main, sebagai contoh berikut.

```
public class Main {
    public static void main(String[] args) {
```

```
        SingleLinkedList<String> list = new
SingleLinkedList<String>();
        System.out.println("SSL isEmpty ? : " + list.isEmpty());
        System.out.println("Size SSL : " + list.size());
        list.add("merah");
        list.add("kuning");
        list.add(0,"ungu");
        list.add("merah");
        list.add("biru");
        System.out.println("List 1: " + list.toString());
        System.out.println("Index 1 : " + list.get(1));
        list.set(0, "ungu 2");
        System.out.println("List 2: " + list.toString());
        System.out.println("Size SSL : " + list.size());
    }
}
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.