

---

# PRAKTIKUM 16

---

## ITERATOR PADA SINGLE LINKED LIST

---

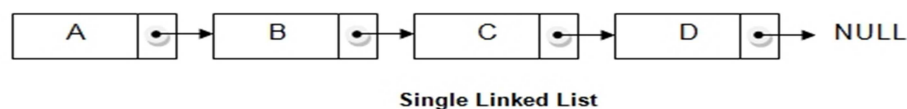
### A. TUJUAN PEMBELAJARAN

Mahasiswa diharapkan mampu :

1. Memahami konsep Iterator pada SingleLinkedList
2. Mengimplementasikan konsep Iterator pada SingleLinkedList

### 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 memiliki 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 3.

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

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

<code>index) {}</code>	pada node pada index tertentu dari SSL	
<code>public T set(int index, T element){}</code>	Method <code>set(index, element)</code> untuk merubah value pada node pada index tertentu dengan element. Method ini mengembalikan value yang lama.	
<code>public String toString() {}</code>	Method <code>toString()</code> untuk mengubah objek SSL menjadi String	
<code>public Object[] toArray() {}</code>	Method <code>toArray()</code> untuk mengubah objek SSL menjadi array dengan tipe Object	<b>SSL 2</b>
<code>public boolean contains(Object o) {}</code>	Method <code>contains()</code> untuk mengecek apakah terdapat node o pada SSL	
<code>public boolean remove(Object o){}</code>	Method <code>remove(Object o)</code> untuk menghapus node o pada SSL	
<code>private T removeFirst() {}</code>	Method <code>removeFirst()</code> untuk menghapus node yang paling depan	
<code>private T removeLast() {}</code>	Method <code>removeLast()</code> untuk menghapus node yang paling akhir	
<code>public T remove(int index){}</code>	Method <code>remove(index)</code> untuk menghapus node pada index tertentu pada SSL	
<code>public int indexOf(Object o){}</code>	Method <code>indexOf(Object o)</code> untuk mendapatkan index pertama kali node o pada SSL	
<code>public int lastIndexOf(Object o){}</code>	Method <code>lastIndexOf(Object o)</code> untuk mendapatkan index terakhir node o pada SSL	
<code>public Iterator&lt;T&gt; iterator() {}</code>	Method <code>iterator()</code> untuk melakukan iterasi pada SSL.	<b>SSL 3</b>
<code>class IteratorImpl implements Iterator&lt;T&gt; { public boolean hasNext() {} public T next() { } public void remove()</code>	Membuat inner class <code>IteratorImpl</code> yang didalamnya terdapat variable dan method. Variabel : <code>Node&lt;T&gt; curr = front;</code> variable <code>curr</code> untuk menandai node yang akan diakses, pengaksesan node dimulai dari front <code>int index = -1 ;</code>	

<pre>{}</pre> <pre>}</pre>	<b>Method:</b> public boolean hasNext(){} : untuk mengecek apakah ada node berikutnya public T next(){} : untuk mengambil node berikutnya public void remove(){}:untuk menghapus node	
----------------------------	---	--

**Percobaan 1.** Membuat inner class IteratorImpl yang didalamnya terdapat method :

public boolean hasNext(){} : untuk mengecek apakah ada node berikutnya

public T next(){} : untuk mengambil node berikutnya

```
public class SingleLinkedList<T> implements List<T>{
...
    class IteratorImpl implements Iterator<T> {

        private Node<T> curr = front;
        int index = -1 ;

        public boolean hasNext() {
            index++ ;
            return index != size;
        }

        public T next() {
            Node<T> prev = curr ;
            curr = curr.next ;
            return prev.nodeValue;
        }
    }
}
```

## E. LATIHAN

Pada inner class IteratorImpl, selesaikan method remove() untuk menghapus node

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

```
public class Main {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.add("hijau");
        list.add("kuning");
        list.add("hijau");
        list.add("biru");
        System.out.println(list.toString());
    }
}
```

```
Iterator it = list.iterator();
while(it.hasNext()){
    String str = (String) it.next();
    if (str.equals("Kuning"))
        it.remove();
}
System.out.println(list.toString());
}
}
```

Output program adalah

[hijau, kuning, hijau, biru]

## F. LAPORAN RESMI

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