# PRAKTIKUM 15 - 16
# SINGLE LINKED LIST

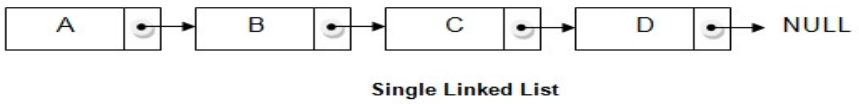
## A. TUJUAN PEMBELAJARAN

Mahasiswa diharapkan mampu :

1. Memahami konsep Linked List
2. Memahami dan mampu membedakan Linked list dengan array
3. Memahami operasi yang ada pada linked list

## B. DASAR TEORI

Linked list adalahsekumpulan 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.

Pembuatan Single Linked List dapat menggunakan 2 metode:

- LIFO (Last In First Out), aplikasinya : Stack (Tumpukan)
- FIFO (First In First Out), aplikasinya : Queue (Antrean)

**Operasi-Operasi yang ada pada Linked List**

- **Insert**

  Istilah Insert berarti menambahkan sebuah simpul baru ke dalam suatu linked list.

- **IsEmpty**

  Fungsi ini menentukan apakah linked list kosong atau tidak.

- **Find First**

  Fungsi ini mencari elemen pertama dari linked list

- **Find Next**

  Fungsi ini mencari elemen sesudah elemen yang ditunjuk now

- **Retrieve**

  Fungsi ini mengambil elemen yang ditunjuk oleh now.Elemen tersebut lalu dikembalikan oleh fungsi.

- **Update**

  Fungsi ini mengubah elemen yang ditunjuk oleh now dengan isi dari sesuatu

- **Delete Now**

  Fungsi ini menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalahelemen pertama dari linked list (head), head akan berpindah ke elemen berikut.

- **Delete Head**

  Fungsi ini menghapus elemen yang ditunjuk head.Head berpindah ke elemen sesudahnya.

- **Clear**

  Fungsi ini menghapus linked list yang sudah ada. Fungsi ini wajib dilakukan bila anda ingin mengakhiri program yang menggunakan linked list. Jika anda melakukannya, data-data yang dialokasikan ke memori pada program sebelumnya akan tetap tertinggal di dalam memori.

## C. TUGAS PENDAHULUAN

Buatlah resume 1 halaman mengenai **Single Linked List** dan berikan penjelasannya.

## D. PERCOBAAN

Node1.java

```
1.  package SingleLinkedList;
2.
3.  import javax.swing.DefaultListModel;
4.
5.  public class Node1 {
6.
7.      public Node1() {
8.      }
9.
10.     public static <T> Node buatLinkedList1(Node<T> front,
    Node<T> p, Node<T> q) {
11.         p = new Node("red");
12.         q = new Node("green");
13.         p.next = q;//pendefinisian q sebagai node setelah
    node terdepan //jika semua node hanya dua dapat diartikan
    bahwa q adalah node paling belakang
14.         front = p;//pendefinisian p sebagai node
    terdepan(front)
15.         return front;//mendefinisikan untuk kembali ke q
    (seperti konsep pointer)
16.     }
17.
18.     public static <T> Node tambahNode_Depan(Node<T> front,
    Node<T> newNode) {
19.         Node temp = front;
20.         if (front == null || front.nodeValue == null) {
21.             front = new Node; //front terdefinisi sebagai
    node baru
22.         } else {
23.             newNode.next = front;
24.         }
25.         front = newNode;
26.         return front;
27.     }
28.
29.     public static <T> Node tambahNode_Akhir(Node<T> front,
    Node<T> newNode) {
30.         Node temp = front;//temp di definisikan sebagai data
    terdepan(front)
31.         if (front.nodeValue == null) {
32.             front = newNode;//jika front sama dengan
    null,maka front akan digantikan node baru
33.         } else {//kondisilainnya
34.             while (temp != null) {//seleksikondisi
35.                 if (temp.next == null) {//seleksi kondisi
    lagi,jika node setelah temp sama dengan null
36.                     temp.next = newNode;//maka tempat setelah
    temp (node akhir) akan diisikan node baru
37.                     break;
38.                 }
39.                 temp = temp.next;
40.             }
41.         }
```

```
42.         return front;
43.     }
44.
45.     public static <T> Node tambahNode_Sebelum(Node<T> front,
    Node<T> newNode, Node<T> curr) {
46.         if (temp.next.nodeValue.equals(curr.nodeValue)) {
47.             temp1 = temp.next;// konsep tidak jauh beda
    dengan proses sebelumnya
48.             temp.next = newNode;//pelajari pointer,untuk
    mengerti alurnya
49.             temp.next.next = temp1;
50.             break;
51.         }
52.         temp = temp.next;
53.     }
54.     return front ;
55.
56.     public static <T extends Comparable<? super T>> Node
    tambahNode_Setelah(Node<T> front, Node<T> newNode, Node<T>
    curr) {
57.         Node temp = front, temp1;
58.         while (temp != null) {
59.             if (temp.nodeValue.equals(curr.nodeValue)) {
60.                 temp1 = temp.next;
61.                 temp.next = newNode;
62.                 temp.next.next = temp1;
63.                 break;
64.             }
65.             temp = temp.next;
66.         }
67.         return front;
68.     }
69.
70.     public static <T> Node hapusNode_Depan(Node<T> front) {
71.         Node temp = front.next;
72.         front = null;
73.         front = temp;
74.         return front;
75.     }
76.
77.     public static <T> Node hapusNode(Node<T> front, Node<T>
    target) {
78.         Node temp = front;
79.         if (front.next == null) {
80.             front.nodeValue = null;
81.         } else {
82.             while (temp != null) {
83.                 if
    (temp.next.nodeValue.equals(target.nodeValue)) {
84.                     temp.next = temp.next.next;
85.                     break;
86.                 }
87. temp = temp.next;
88.             }
89.         }
```

```
90.          return front;
91.      }
92.
93.      public static <T> DefaultListModelgetData(Node<T> front)
    {
94.          DefaultListModel model = new DefaultListModel();
95.          inti = 0;
96.          if (front == null) {
97.              return null;
98.          }
99.          Node<T> curr = front;
100.           model.addElement(curr.nodeValue);
101.           while (curr.next != null) {
102.               i++;
103.               curr = curr.next;
104.               model.addElement(curr.nodeValue);
105.           }
106.
107.           return model;
108.       }
109.
110.       public static <T> String toString(Node<T> front) {
111.           if (front == null) {
112.               return "null";
113.           }
114.           Node<T> curr = front;
115.           java.lang.Stringstr = "" + curr.nodeValue;
116.           while (curr.next != null) {
117.               curr = curr.next;
118.               str += "" + curr.nodeValue;
119.           }
120.           str += "";
121.           return str;
122.       }
123.   }
```

LinkedListString.java

```
1.    import javax.swing.JOptionPane;
2.
3.    public class LinkedListString extends javax.swing.JFrame {
4.
5.        Node front = new Node();
6.
7.        publicLinkedListString() {
8.            initComponents();
9.            popup.resize(150, 150);
10.           tambah.resize(150, 150);
11.       }
12.
13.       private void
      AddActionPerformed(java.awt.event.ActionEventevt) {
14.           popup.setVisible(true);
15.       }
16.
```

```
17.      private void
     HasilInputMethodTextChanged(java.awt.event.InputMethodEvent
     evt) {
18.          System.out.println(evt.toString());
19.      }
20.
21.      private void
     hapusbtnActionPerformed(java.awt.event.ActionEventevt) {
22.          try {
23.              front = Node1.hapusNode(front, new
     Node(Hasil.getSelectedValue()));
24.              Hasil.setModel(Node1.getData(front));
25.              System.out.println("data awal \n" +
     Node1.toString(front));
26.          } catch (Exception e) {
27.          }
28.      }
29.
30.      private void
     tambahdepanActionPerformed(java.awt.event.ActionEventevt) {
31.          try {
32.              String str =
     JOptionPane.showInputDialog("MasukanKalimat");
33.              front = Node1.tambahNode_Depan(front, new
     Node(str));
34.              Hasil.setModel(Node1.getData(front));
35.              System.out.println("data awal \n" +
     Node1.toString(front));
36.          } catch (Exception e) {
37.          }
38.      }
39.
40.      private void
     tambahakhirActionPerformed(java.awt.event.ActionEventevt) {
41.          try {
42.              String str =
     JOptionPane.showInputDialog("MasukanKalimat");
43.              front = Node1.tambahNode_Akhir(front, new
     Node(str));
44.              Hasil.setModel(Node1.getData(front));
45.              System.out.println("data awal\n" +
     Node1.toString(front));
46.          } catch (Exception e) {
47.          }
48.      }
49.
50.      private void
     tmbsblmActionPerformed(java.awt.event.ActionEventevt) {
51.          try {
52.              String str1 =
     JOptionPane.showInputDialog("PilihKalimat yang akan di
     inputkan ");
53.              String str =
     JOptionPane.showInputDialog("InputkanKalimatnya : ");
54.              front = Node1.tambahNode_Sebelum(front, new
```

```
      Node(str), new Node(str1));
55.               Hasil.setModel(Node1.getData(front));
56.               System.out.println("data awal \n" +
      Node1.toString(front));
57.           } catch (Exception e) {
58.           }
59.       }
60.
61.       private void
      tmbstlahActionPerformed(java.awt.event.ActionEventevt) {
62.           try {
63.               String str1 =
      JOptionPane.showInputDialog("PilihKalimat");
64.               String str =
      JOptionPane.showInputDialog("InputkanKalimatnya : ");
65.               front = Node1.tambahNode_Setelah(front, new
      Node(str), new Node(str1));
66.               Hasil.setModel(Node1.getData(front));
67.               System.out.println("data awal \n" +
      Node1.toString(front));
68.           } catch (Exception e) {
69.           }
70.       }
71.
72.       private void
      insertbtnActionPerformed(java.awt.event.ActionEventevt) {
73.           tambah.setVisible(true);
74.       }
75.
76.       private void
      jBExitActionPerformed(java.awt.event.ActionEventevt) {
77.           System.exit(1);
78.       }
79.
80.       public static void main(String args[]) {
81.           java.awt.EventQueue.invokeLater(new Runnable() {
82.               @Override
83.               public void run() {
84.                   newLinkedListString().setVisible(true);
85.               }
86.           });
87.       }
88.       // Variables declaration - do not modify
89.       privatejavax.swing.JButton Add;
90.       privatejavax.swing.JListHasil ;
91.       privatejavax.swing.JButtonhapusbtn ;
92.       privatejavax.swing.JButtoninsertbtn ;
93.       privatejavax.swing.JButtonjBExit ;
94.       privatejavax.swing.JLabel jLabel1;
95.       privatejavax.swing.JScrollPane jScrollPane1;
96.       privatejavax.swing.JDialog popup;
97.       privatejavax.swing.JDialogtambah ;
98.       privatejavax.swing.JButtontambahakhir ;
99.       privatejavax.swing.JButtontambahdepan ;
100.      privatejavax.swing.JButtontmbsblm ;
```

```
101.      privatejavax.swing.JButtontmbstlah ;
102.      // End of variables declaration
103.  }
```

## E. LATIHAN

1. Buatlah sebuah class Node yang merepresentasikan Single Linked List yang memiliki atribut
   - nodeValue : untuk menyimpan data/value dari Node
   - next : untuk menunjuk ke node berikutnya

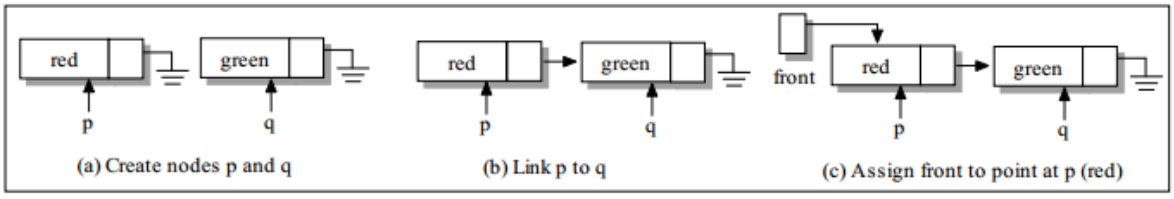
```
1.  public class Node<T> {
2.     …
3.     // default constructor with no initial value
4.     public Node() { …
5.
6.     }
7.     // initialize nodeValue to item and set next to null
8.     public Node(T item) { …
9.
10.    }
11. }
```

Buatlah class SingleLinkedList yang didalamnya terdapat fungsi-fungsi sebagai berikut.

2. Buatlah fungsi buatLinkedList1() untuk membuat LinkedList seperti gambar 1public static <T> Node buatLinkedList1(Node<T> front, Node<T> p, Node<T> q)
   a. Buatlah link dari p ke q.
   b. Buatlah objek front dan arahkan front ke p.

   Sebelum memanggil fungsi buatLinkedList1() lakukan :

   Buatlah objek p dan q dari class Node. Objek p nodeValue : red, objek q berisi nodeValue : green.

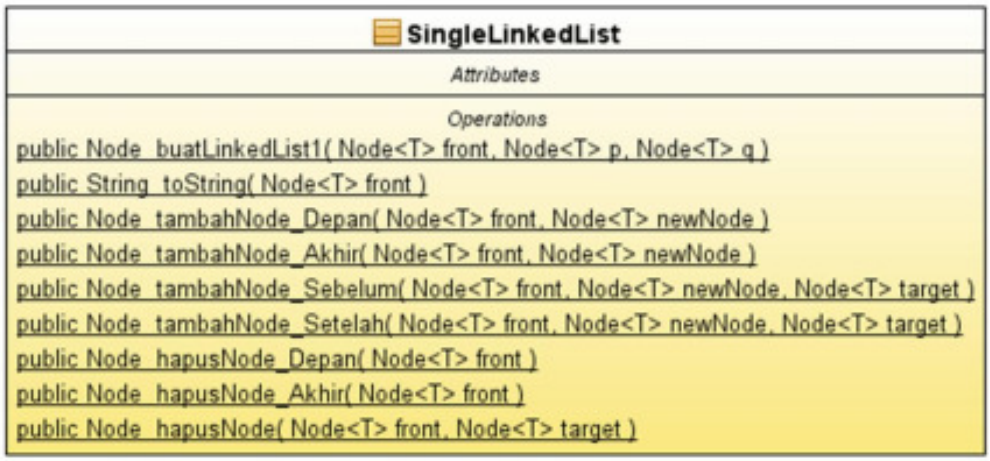



Gambar 2. Soal Single Linked List

3. Buatlah fungsi **toString()** untuk melakukan pembacaan Linked List

   **public static <T> String toString(Node<T> front)**

4. Buatlah fungsi untuk menambahkan Node di awal Linked List
   **public static <T> Node tambahNode_Depan(Node<T> front, Node<T> newNode)**
5. Buatlah fungsi untuk menambahkan Node di akhir Linked List
   **public static <T> Node tambahNode_Akhir(Node<T> front, Node<T> newNode)**
6. Buatlah fungsi untuk menambahkan Node di tengah Linked List, sebelum Node tertentu
   **public static <T> Node tambahNode_Sebelum(Node<T> front, Node<T> newNode, Node<T> curr).**
7. Buatlah fungsi untuk menambahkan Node di tengah Linked List, setelah Node tertentu
   **public static <T> Node tambahNode_Setelah(Node<T> front, Node<T> new Node, Node<T> curr).**
8. Buatlah fungsi untuk menghapus Node di awal Linked List
   **public static <T> Node hapusNode_Depan(Node<T> front).**
9. Buatlah fungsi untuk menghapus Node di akhir Linked List
   **public static <T> Node hapusNode_Akhir(Node<T> front).**
10. Buatlah fungsi untuk menghapus Node tertentu
    **public static <T> Node hapusNode (Node<T> front, Node<T> target ).**




Gambar 3. Class SingleLinkedList

Selanjutnya buatlah class **TestSingleLinkedList** untuk menguji fungsi-fungsi tersebut dan jangan lupa untuk menampilkan (dengan memanggil fungsi toString()) !

```
public class TestSingleLinkedList {
    public static void main(String[] args) {
        //A2
        Node<String> front = null;
```

```
        front    =    SingleLinkedList.buatLinkedList1(front,    new
Node<String>("red"), new Node<String>("green"));
        System.out.println(SingleLinkedList.toString(front));
        //A4
        front    =    SingleLinkedList.tambahNode_Depan(front,    new
Node<String>("black"));
        System.out.println(SingleLinkedList.toString(front));
        //A5
        front    =    SingleLinkedList.tambahNode_Akhir(front,    new
Node<String>("YELLOW"));
        System.out.println(SingleLinkedList.toString(front));
        //A6 - depan
        front   =   SingleLinkedList.tambahNode_Sebelum(front,   new
Node<String>("purple"), new Node<String>("black"));
        System.out.println(SingleLinkedList.toString(front));
        //A6 - tengah
        front   =   SingleLinkedList.tambahNode_Sebelum(front,   new
Node<String>("orange"), new Node<String>("red"));
        System.out.println(SingleLinkedList.toString(front));
        //A6 - belakang
        front   =   SingleLinkedList.tambahNode_Sebelum(front,   new
Node<String>("jingga"), new Node<String>("green"));
        System.out.println(SingleLinkedList.toString(front));
        //dst
    }
}
```

11. Amati listing berikut ini, dan bagaimana outputnya?

```
import java.util.*;

publicclassLinkedListDemo{

publicstaticvoid main(String args[]){
// create a linked list
LinkedList ll =newLinkedList();
// add elements to the linked list
      ll.add("F");
      ll.add("B");
      ll.add("D");
      ll.add("E");
      ll.add("C");
      ll.addLast("Z");
      ll.addFirst("A");
      ll.add(1,"A2");
System.out.println("Original contents of ll: "+ ll);

// remove elements from the linked list
      ll.remove("F");
      ll.remove(2);
System.out.println("Contents of ll after deletion: "
+ ll);

// remove first and last elements
      ll.removeFirst();
      ll.removeLast();
```

```
System.out.println("ll after deleting first and last: "
+ ll);

// get and set a value
Object val = ll.get(2);
      ll.set(2,(String) val +" Changed");
System.out.println("ll after change: "+ ll);
}
}
```

## F. LAPORAN RESMI

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