



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

## PERCOBAAN 1

```
1 package P11.StrukturData;  
2  
3 public class Node {  
4     int data;  
5     Node next;  
6  
7     public Node(int nilai, Node berikutnya) {  
8         this.data = nilai;  
9         this.next = berikutnya;  
10    }  
11  
12 }  
13
```

```
1 package P11.StrukturData;  
2  
3 public class SLLMain {  
4     public static void main(String[] args) {  
5         SingleLinkedList singLL = new SingleLinkedList();  
6  
7         singLL.print();  
8         singLL.addFirst(890);  
9         singLL.print();  
10        singLL.addlast(760);  
11        singLL.print();  
12        singLL.addFirst(700);  
13        singLL.print();  
14        singLL.insertAfter(700, 999);  
15        singLL.print();  
16        singLL.insertAt(3, 833);  
17        singLL.print();  
18    }  
19 }  
20
```



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

```
1 package p11.StrukturData;
2
3 public class SingleLinkedList {
4     Node head, tail;
5
6     boolean isEmpty() {
7         return head == null;
8     }
9
10    void print() {
11        if (!isEmpty()) {
12            Node tmp = head;
13            System.out.print("Isi Linked List: ");
14            while (tmp != null) {
15                System.out.print(tmp.data + "\t");
16                tmp = tmp.next;
17            }
18            System.out.println("");
19        } else {
20            System.out.println("Linked List kosong");
21        }
22    }
23
24    void addFirst(int input) {
25        Node ndInput = new Node(input, head);
26
27        if (isEmpty()) {
28            head = ndInput;
29            tail = ndInput;
30        } else {
31            ndInput.next = head;
32            head = ndInput;
33        }
34    }
35
36    void addlast(int input) {
37        Node ndInput = new Node(input, null);
38        if (isEmpty()) {
39            head = ndInput;
40            tail = ndInput;
41        } else {
42            tail.next = ndInput;
43            tail = ndInput;
44        }
45    }
46
47    void insertAfter(int key, int input) {
48        Node ndInput = new Node(input, null);
49        Node temp = head;
50        do {
51            if (temp.data == key) {
52                ndInput.next = temp.next;
53                temp.next = ndInput;
54                if (ndInput.next == null) {
55                    tail = ndInput;
56                    break;
57                }
58            }
59            temp = temp.next;
60        } while (temp != null);
61    }
62
63    void insertAt(int index, int input) {
64        if (index < 0) {
65            System.out.println("Indeks salah");
66        } else if (index == 0) {
67            addFirst(input);
68        } else {
69            Node temp = head;
70            for (int i = 0; i < index - 1; i++) {
71                if (temp.next == null)
72                    break;
73                temp = temp.next;
74            }
75            temp.next = new Node(input, temp.next);
76            if (temp.next.next == null) {
77                tail = temp.next;
78            }
79        }
80    }
81 }
82 }
83 }
```

```
Linked List kosong
Isi Linked List: 890
Isi Linked List: 890    760
Isi Linked List: 700    890    760
Isi Linked List: 700    999    890    760
Isi Linked List: 700    999    890    833    760
```



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

## Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?

Jawab:

```
singLL.print();  
singLL.addFirst(input:890);  
singLL.print();  
singLL.addlast(input:760);  
singLL.print();  
singLL.addFirst(input:700);  
singLL.print();  
singLL.insertAfter(key:700, input:999);  
singLL.print();  
singLL.insertAt(index:3, input:833);  
singLL.print();
```

Baris pertama menghasilkan “Linked List Kosong” karena pada linked list tersebut belum diisi sebuah data dan pemanggilan print dilakukan sebelum pengisian data sehingga output program baris pertama adalah “Linked List Kosong”.

2. Jelaskan kegunaan variable temp secara umum pada setiap method!

Jawab:

Variabel temp digunakan untuk menampung node sementara.

3. Perhatikan class SingleLinkedList, pada method insertAt Jelaskan kegunaan kode berikut

```
if(temp.next.next==null) tail=temp.next;
```

Jawab:

Code tersebut adalah pengecekan kondisi yang digunakan ketika kondisi di dalam if terpenuhi maka nilai tail akan diperbarui menjadi temp.next



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

## PERCOBAAN 2

```
1 package P11.StrukturData;
2
3 public class SLLMain {
4     public static void main(String[] args) {
5         SingleLinkedList singLL = new SingleLinkedList();
6
7         singLL.print();
8         singLL.addFirst(890);
9         singLL.print();
10        singLL.addlast(760);
11        singLL.print();
12        singLL.addFirst(700);
13        singLL.print();
14        singLL.insertAfter(700, 999);
15        singLL.print();
16        singLL.insertAt(3, 833);
17        singLL.print();
18
19        System.out.println("Data pada indeks ke-1 = " + singLL.getData(1));
20        System.out.println("Data 3 berada pada indeks ke-" + singLL.indexOf(760));
21
22        singLL.remove(999);
23        singLL.print();
24        singLL.removeAt(0);
25        singLL.print();
26        singLL.removeFirst();
27        singLL.print();
28        singLL.removeLast();
29        singLL.print();
30    }
31 }
32
```

Linked List kosong

Isi Linked List: 890

Isi Linked List: 890      760

Isi Linked List: 700      890      760

Isi Linked List: 700      999      890      760

Isi Linked List: 700      999      890      833      760

Data pada indeks ke-1 = 999

Data 3 berada pada indeks ke-4

Isi Linked List: 700      890      833      760

Isi Linked List: 890      833      760

Isi Linked List: 833      760

Isi Linked List: 833



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

```
1 package M1.StrukturData;
2
3 public class SingleLinkedList {
4     Node head, tail;
5
6     boolean isEmpty() {
7         return head == null;
8     }
9
10    void print() {
11        if (isEmpty()) {
12            Node temp = head;
13            System.out.println("List linked list: ");
14            while (temp != null) {
15                System.out.print(temp.data + " ");
16                temp = temp.next;
17            }
18            System.out.println("");
19        } else {
20            System.out.println("Linked list kosong");
21        }
22    }
23
24    void addFirst(int input) {
25        Node newNode = new Node(input, head);
26
27        if (isEmpty()) {
28            head = newNode;
29            tail = newNode;
30        } else {
31            newNode.next = head;
32            head = newNode;
33        }
34    }
35
36    void addLast(int input) {
37        Node newNode = new Node(input, null);
38        if (isEmpty()) {
39            head = newNode;
40            tail = newNode;
41        } else {
42            tail.next = newNode;
43            tail = newNode;
44        }
45    }
46
47    void insertAfter(int key, int input) {
48        Node newNode = new Node(input, null);
49        Node temp = head;
50        do {
51            if (temp.data == key) {
52                newNode.next = temp.next;
53                temp.next = newNode;
54                if (temp.next == null) {
55                    tail = newNode;
56                }
57            }
58            temp = temp.next;
59        } while (temp != null);
60    }
61
62    void insertAt(int index, int input) {
63        if (index < 0) {
64            System.out.println("Index salah");
65        } else if (index == 0) {
66            addFirst(input);
67        } else {
68            Node temp = head;
69            for (int i = 0; i < (index - 1); i++) {
70                if (temp.next == null) {
71                    break;
72                }
73                temp = temp.next;
74            }
75            temp.next = new Node(input, temp.next);
76            if (temp.next.next == null) {
77                tail = temp.next;
78            }
79        }
80    }
81
82    public int getAt(int index) {
83        Node temp = head;
84        for (int i = 0; i < index; i++) {
85            temp = temp.next;
86        }
87        return temp.data;
88    }
89
90    public int indexOf(int key) {
91        int index = 0;
92        while (temp != null && temp.data != key) {
93            temp = temp.next;
94            index++;
95        }
96        if (temp == null) {
97            return -1;
98        } else {
99            return index;
100        }
101    }
102
103    public void removeFirst() {
104        if (isEmpty()) {
105            System.out.println("Linked list masih kosong, tidak dapat dihapus");
106        } else if (head == tail) {
107            head = tail = null;
108        } else {
109            head = head.next;
110        }
111    }
112
113    public void removeLast() {
114        if (isEmpty()) {
115            System.out.println("Linked list masih kosong, tidak dapat dihapus");
116        } else if (head == tail) {
117            head = tail = null;
118        } else {
119            Node temp = head;
120            while (temp.next != tail) {
121                temp = temp.next;
122            }
123            temp.next = null;
124            tail = temp;
125        }
126    }
127
128    public void remove(int key) {
129        if (isEmpty()) {
130            System.out.println("Linked list masih kosong, tidak dapat dihapus");
131        } else {
132            Node temp = head;
133            while (temp != null) {
134                if ((temp.data == key) && (temp == head)) {
135                    removeFirst();
136                } else if ((temp.data == key) && (temp != head)) {
137                    temp.next = temp.next.next;
138                    if (temp.next == null) {
139                        tail = temp;
140                    }
141                }
142                temp = temp.next;
143            }
144        }
145    }
146
147    public void removeAll() {
148        if (isEmpty()) {
149            System.out.println("Index salah");
150        } else if (index == 0) {
151            removeFirst();
152        } else {
153            Node temp = head;
154            for (int i = 0; i < (index - 1); i++) {
155                if (temp.next == null) {
156                    break;
157                }
158                temp = temp.next;
159            }
160            temp.next = temp.next.next;
161            if (temp.next == null) {
162                tail = temp;
163            }
164        }
165    }
166 }
```



NAMA : Roy Wijaya  
NIM : 2341720120  
KELAS : 1-G  
MATERI : Linked List (Jobsheet 9)

## PERTANYAAN

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!

Jawab:

Keyword break pada fungsi remove digunakan untuk menghentikan iterasi, karena jika dihapus, iterasi akan terus berjalan walaupun node yang dicari sudah ditemukan

2. Jelaskan kegunaan kode dibawah pada method remove

```
else if (temp.next.data == key) {  
    temp.next = temp.next.next;
```

Jawab:

Kode tersebut digunakan untuk pengecekan kondisi Ketika node yang dicari sama dengan node saat ini (temp.next). Kemudian kode tersebut melakukan penghapusan node dengan cara mengubah pointer dari node temp langsung menuju ke node yang berada setelah node yang akan dihapus.

LINK GITHUB:

[https://github.com/RoyW12/AlgoritmaStrukturData\\_1G\\_28/tree/main/src/P11/StrukturData](https://github.com/RoyW12/AlgoritmaStrukturData_1G_28/tree/main/src/P11/StrukturData)