



Teknik Informatika Kampus 3 Nganjuk  
Jurusan Teknologi Informasi

Politeknik Negeri  
Jember

# Program Single Linked List

Struktur Data

Ulfa Emi Rahmawati, S.Kom., M.Kom.

## Pengertian

Membahas bagaimana membuat program Single Linked List, terbagi menjadi :

- Inisiasi Node
- Insertion, penyisipan suatu node
  - Di awal Node, di akhir Node, dan di Tengah-tengah
- Deletion, penghapusan suatu node
  - Di awal Node, di akhir Node, dan di Tengah-tengah
- Menampilkan Node

Single Linked List dibuat menggunakan Bahasa Pemrograman Java.

# Inisiasi Node

Cara menginisiasi Node

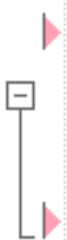
## Inisiasi Node

Sebuah Node diinisiasi:

- Berbentuk Class
- Atribut berupa
  - **data**, atribut untuk menampung nilai dari suatu Node
  - **next**, sebagai penunjuk node selanjutnya.
- Constructor, method dipanggil pertama kali ketika membuat objek baru
  - Berisi parameter **data** memanggil atribut kelas itu sendiri
  - **Next** yang bernilai **null**.

## Inisiasi Node

```
12 public class Node {  
13     int data;  
14     Node next;  
15  
16     public Node(int data) {  
17         this.data = data;  
18         next = null;  
19     }  
20
```



# Insertion

Penyisipan Node baru

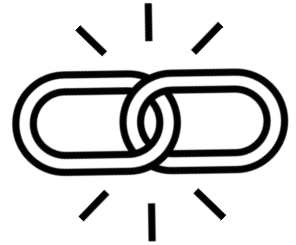
# Insertion

Merupakan penyisipan node baru pada suatu linked list

- Insertion di awal
- Insertion di akhir
- Insertion setelah node tertentu atau ditengah linked list

# Perhatikan!

Yang menjadi perhatian pada suatu linked list adalah **Link**-nya (tautannya)!





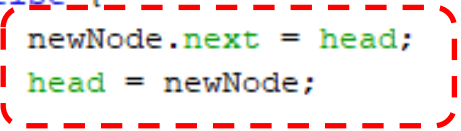

## Insertion di Awal

Merupakan penyisipan node baru di awal node

- Buat class baru Bernama **SingleLinkedList**
- Inisiasi Node diawal sebagai **null**
- Buat method **addAtTheFront** dengan parameter data dan nilai kembalian **Node**
- Periksa, jika node masih kosong maka node yang ditambahkan akan menjadi node baru
- Jika sudah ada, arahkan nilai **next** pada node baru sebagai **head**
- Simpan node baru sebagai head

## Insertion di Awal

```
14 public class SingleLinkedList {
15
16     Node head = null;
17
18     //insertion at the front
19     public Node addAtTheFront(int data) {
20         System.out.println("Add a Node " + data + " at the beginning");
21         Node newNode = new Node(data);
22         if (head == null) {
23             head = newNode;
24         } else {
25             newNode.next = head;
26             head = newNode;
27         }
28         return newNode;
29     }
```



sintaks untuk mengarah/menunjuk  
pada link selanjutnya

## Insertion di Akhir

Merupakan penyisipan node baru di akhir node

- Buat method **addAtTheEnd** dengan parameter data dan nilai kembalian **Node**
- Periksa, jika node masih kosong maka node yang ditambahkan akan menjadi node baru
- Jika sudah ada, atur node ke-1 sebagai **head**
- Carilah **node** selanjutnya (**next**) dengan syarat node selanjutnya tidak sama dengan **null**.
- Jika ditemukan **node.next** bernilai **null** (menunjuk ke null), maka sisipkan **node baru** pada **node.next** (**node.next** mengarah ke node baru)

## Insertion di Akhir

```
31 //insertion at the end
32 public Node addAtTheEnd(int data) {
33     System.out.println("Add a Node " + data + " at the end");
34     Node newNode = new Node(data);
35     if (head == null) {
36         head = newNode;
37     } else {
38         Node currentNode = head;
39         while (currentNode.next != null) {
40             currentNode = currentNode.next;
41         }
42         currentNode.next = newNode;
43     }
44     return newNode;
45 }
```

sintaks mencari node.next yang menunjuk ke null

} sintak penyisipan node baru setelah node.next

## Insertion setelah Node Tertentu

Merupakan penyisipan node baru setelah node tertentu

- Buat method **addNodeAfter** dengan parameter **after data**, data dan nilai kembalian **Node**
- Periksa, jika node masih kosong maka beri notifikasi
- Jika sudah ada, atur node ke-1 sebagai node yang dicari (**searchedNode**). Kemudian pindah pencarian ke **node** selanjutnya dengan cara **memindah** `currentNode.next` menjadi `currentNode`.
- **Bandingkan**, jika pada **searchedNode.data** sama dengan node yang dicari (**after data**), maka sisipkan node baru pada **searchedNode.data**

## Insertion setelah Node Tertentu

- Kemudian pada **newNode.next**, **sisipkan** (tautkan/link) **currentNode**.

## Insertion setelah Node tertentu

```
47 public Node addNodeAfter(int afterData, int data) {  
48     Node newNode = new Node(data);  
49     Node currentNode = head, searchedNode;  
50     System.out.println("Add a Node "+data+" after "+afterData);  
51     if (head == null) {  
52         System.out.println("Linked list is empty");  
53     } else {  
54         while (currentNode != null) {  
55             searchedNode = currentNode;  
56             currentNode = currentNode.next;  
57             if (searchedNode.data == afterData && currentNode != null) {  
58                 searchedNode.next = newNode;  
59                 newNode.next = currentNode;  
60             }  
61         }  
62     }  
63     return currentNode;  
64 }
```

sintaks menggeser node ke selanjutnya

Penyisipan newnode setelah after data

sintak penunjukkan newNode.next ke node sekarang

# Cetak Linked List

Mencetak linked list yang terbentuk



## Cetak linked list

```
66 public void printLinkedList() {  
67     System.out.println("\n == Print Linked List ==");  
68     if (head == null) {  
69         System.out.println("Linked list is empty");  
70     } else {  
71         System.out.print("(HEAD) ");  
72         Node currentNode = head;  
73         while (currentNode != null) {  
74             System.out.print(currentNode.data + "->");  
75             currentNode = currentNode.next;  
76         }  
77         System.out.print("NULL\n");  
78     }  
79 }
```

Atur head sebagai currentNode

Cetak data pada current Node

Geser node selanjutnya sebagai currentNode

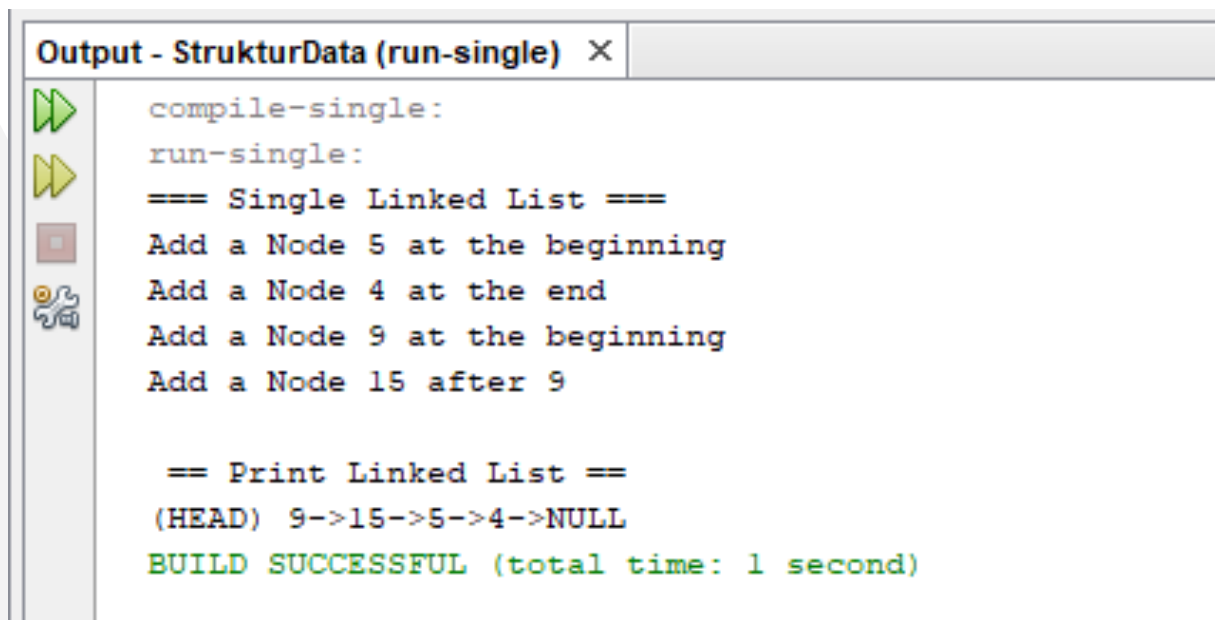
# Membuat linked list

Membuat linked list pada method main

## Insertion di Akhir

```
13 public class Main {  
14     public static void main(String[] args) {  
15  
16         System.out.println("=== Single Linked List ===");  
17         SingleLinkedList single = new SingleLinkedList();  
18         single.addAtTheFront(5);  
19         single.addAtTheEnd(4);  
20         single.addAtTheFront(9);  
21         single.addNodeAfter(9, 15);  
22         single.printLinkedList();  
23     }  
24 }  
25  
26
```

# Output Program



```
Output - StrukturData (run-single) X
compile-single:
run-single:
=== Single Linked List ===
Add a Node 5 at the beginning
Add a Node 4 at the end
Add a Node 9 at the beginning
Add a Node 15 after 9

== Print Linked List ==
(HEAD) 9->15->5->4->NULL
BUILD SUCCESSFUL (total time: 1 second)
```

# THANKS!

**Any questions?**