

- **NAMA** : CHANDRA HARKAT RAHARJA
NPM : 233040089
KELAS : PRAKTIKUM PEMROGRAMAN I (B)
GITHUB : [PP12025_B_233040089/TugasDanPertemuanCoding/TugasPertemuan3 at main · ComradeChandra/PP12025_B_233040089](https://github.com/PP12025_B_233040089/TugasDanPertemuanCoding/TugasPertemuan3_at_main·ComradeChandra/PP12025_B_233040089)

Latihan-1

Latihan 1

Latihan ini akan memberikan implementasi pembuatan elemen list. Elemen list direpresentasikan dengan Node. Sebuah node terdiri dari atribut data/nilai dan atribut *next*. Atribut *next* akan menunjuk ke node yang lain.

Pseudocode	Bahasa Pemrograman
<pre>class Node { data: tipe data (T) next: Node }</pre>	<pre>public class Node { private int data; private Node next; /** Inisialisasi atribut node */ public Node(int data) { this.data = data; } /** Setter & Getter */ }</pre>

- a. Contoh setter: `public void setName(String nama) { this.nama = nama; }`
b. Contoh getter: `public String getName() { return nama; }`

```
TugasDanPertemuanCoding > TugasPertemuan3 > Node.java > Node
1  public class Node {
2      private int data;
3      private Node next;
4
5      // Inisialisasi Atribut Node //
6      public Node(int data) {
7          this.data = data;
8      }
9
10     // Setter&Getter //
11
12     public int getData() {
13         return data;
14     }
15
16     public void setData(int data) {
17         this.data = data;
18     }
19
20     public Node getNext() {
21         return next;
22     }
23
24     public void setNext(Node next) {
25         this.next = next;
26     }
27
28
29 }
```

Penjelasan:

Deklarasi Kelas dan Atribut

- `public class Node {`
Mendefinisikan kelas dengan nama "Node" yang bersifat publik, sehingga dapat diakses dari kelas lain.
- `private int data;`
Variabel privat "data" bertipe integer digunakan untuk menyimpan nilai atau informasi pada node.
- `private Node next;`
Variabel privat "next" bertipe Node menyimpan referensi ke node berikutnya. Ini memungkinkan kita menghubungkan node satu dengan node lainnya membentuk linked list.

Konstruktor

- `public Node(int data) {`
Konstruktor ini dipanggil saat kita membuat objek Node baru. Parameter "data" digunakan untuk menginisialisasi nilai yang ingin disimpan.
- `this.data = data;`
Baris ini mengassign nilai dari parameter ke variabel instance "data".
(Catatan: Variabel "next" tidak diinisialisasi di sini, sehingga secara default bernilai null, yang berarti node ini belum terhubung ke node lain.)

Getter dan Setter untuk Atribut

- Getter untuk data:
`public int getData() { return data; }`
Method ini mengembalikan nilai yang tersimpan dalam variabel "data", sehingga bisa diakses dari luar kelas.
- Setter untuk data:
`public void setData(int data) { this.data = data; }`
Method ini mengatur atau mengganti nilai dalam variabel "data" dengan nilai baru yang diberikan.
- Getter untuk next:
`public Node getNext() { return next; }`
Method ini mengembalikan referensi ke node berikutnya, sehingga kita dapat mengetahui node mana yang terhubung setelah node ini.
- Setter untuk next:
`public void setNext(Node next) { this.next = next; }`
Method ini digunakan untuk menetapkan referensi ke node berikutnya, menghubungkan node ini dengan node lain dalam linked list.

-Latihan 2

Latihan 2

Latihan ini akan memberikan implementasi operasi penambahan elemen list di akhir/*tail* dengan notasi algoritma. Operasi ini direpresentasikan dengan fungsi **addTail** dengan parameter data yaitu node yang akan ditambahkan ke List.

- Buatlah kelas **StrukturList** kemudian tambahkan atribut HEAD dengan tipe data Node
- Tambahkan fungsi dibawah ini di kelas **StrukturList**. Fungsi addTail di bawah dikonversi ke dalam bahasa pemrograman

Algoritma addTail	Program addTail
<pre>procedure addTail(data: integer) deklarasi posNode, curNode: Node {current node} deskripsi newNode ← new Node(data) IF (isEmpty()) THEN HEAD ← newNode ELSE curNode ← HEAD WHILE (curNode <> null) DO posNode ← curNode curNode ← curNode.next ENDWHILE posNode.next ← newNode ENDIF</pre>	<pre>public void addTail(int data) { Node posNode=null, curNode=null; Node newNode = new Node(data); if (isEmpty()) { HEAD = newNode; } else { curNode = HEAD; while (curNode != null) { posNode = curNode; curNode = curNode.getNext(); } posNode.setNext(newNode); } }</pre>

- *function isEmpty()* harus diimplementasikan dengan melakukan apakah list kosong atau tidak (HEAD != null)!

TugasDanPertemuanCoding > TugasPertemuan3 > StrukturList.java > StrukturList > display()

```
1  public class StrukturList {
2
3      Node HEAD;
4
5      public boolean isEmpty(){
6          return (HEAD == null);
7      }
8
9      public void addTail(int data){
10         Node posNode=null, curNode=null;
11         Node newNode = new Node(data);
12         if (isEmpty()){
13             HEAD = newNode;
14         }else{
15             curNode = HEAD;
16             while (curNode != null){
17                 posNode = curNode;
18                 curNode = curNode.getNext();
19             }
20             posNode.setNext(newNode);
21         }
22     }
```

Penjelasan:

➤ Deklarasi Kelas dan Atribut

- `public class StrukturList { ... }`
Kelas StrukturList digunakan untuk menyimpan dan mengelola node-node dalam linked list.
- Node HEAD;
Variabel HEAD bertipe Node menandakan awal (head) dari linked list. Jika HEAD bernilai null, berarti list masih kosong.

➤ Metode isEmpty()

- `public boolean isEmpty() { return (HEAD == null); }`
Metode ini memeriksa apakah linked list kosong dengan melihat apakah HEAD bernilai null.
 - ❖ Jika `HEAD == null`, maka list kosong dan metode mengembalikan true.
 - ❖ Jika `HEAD != null`, maka list tidak kosong dan metode mengembalikan false.

➤ Metode addTail(int data)

- `public void addTail(int data) { ... }`
Metode ini menambahkan node baru di bagian akhir (tail) dari linked list.
- `Node posNode = null, curNode = null;`
Variabel posNode dan curNode dideklarasikan untuk membantu penelusuran hingga ke ujung list.
- `Node newNode = new Node(data);`
Membuat node baru dengan data yang diterima sebagai parameter.
- `if (isEmpty()) { HEAD = newNode; } else { ... }`
 - ❖ Jika list masih kosong (`isEmpty()` bernilai true), maka node baru langsung menjadi HEAD.
 - ❖ Jika list tidak kosong, penelusuran dilakukan untuk mencari node terakhir.
- `curNode = HEAD; while (curNode != null) { posNode = curNode; curNode = curNode.getNext(); }`
 - ❖ curNode diawali dari HEAD.
 - ❖ Selama curNode tidak null, simpan curNode saat ini ke posNode, lalu bergerak ke node berikutnya.
 - ❖ Ketika perulangan berakhir, posNode akan berisi node terakhir di list.
- `posNode.setNext(newNode);`
 - ❖ Menghubungkan node terakhir (posNode) dengan node baru (newNode).
 - ❖ Dengan demikian, node baru ditempatkan di posisi paling akhir (tail).

-Latihan 3 dan Latihan 4

Latihan 3

Latihan ini akan memberikan implementasi untuk menampilkan elemen list. Elemen list yang ditampilkan ke layar diawali dari nilai HEAD.

Algoritma	Bahasa Pemrograman
<pre>procedure displayElement() deklarasi curNode: Node deskripsi curNode ← HEAD; WHILE (curNode <> null) DO print(curNode.data) curNode ← curNode.next ENDDO</pre>	<pre>public void displayElement() { Node curNode = HEAD; while (curNode != null) { System.out.print(curNode.getData() + " "); curNode = curNode.getNext(); } }</pre>

Latihan 4

Latihan ini akan memberikan penggunaan operasi penambahan elemen di akhir list dan kemudian menampilkan setiap elemen yang terdapat di list. Buatlah kelas **ListTest** berikut fungsi main untuk mengeksekusi program. Konversikan urutan instruksi berikut di bawah ini ke fungsi tersebut!

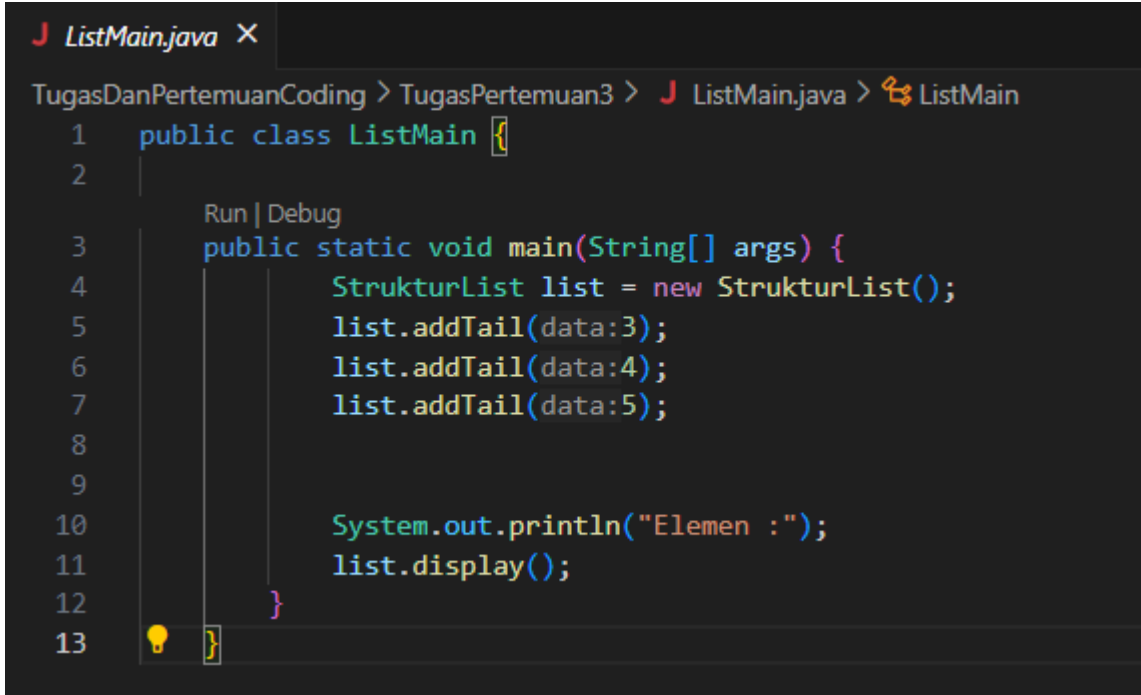
Urutan Instruksi	Program
<ol style="list-style-type: none">1. Create list dengan keyword new2. Tambah elemen 3 di akhir list3. Tambah elemen 4 di akhir list4. Tambah elemen 5 di akhir list5. Tampilkan elemen list	<pre>public class ListMain { public static void main(String[] args) { StrukturList list = new StrukturList(); list.addTail(3); list.addTail(4); list.addTail(5); System.out.println("Elemen: "); list.display(); } }</pre>

Outputnya adalah: **Elemen: 3 4 5**

```
//Latihan 3//
public void display(){
  Node curNode = HEAD;
  while (curNode != null){
    System.out.print(curNode.getData() + " ");
    curNode = curNode.getNext();
  }
}
```

Penjelasan:

- Node curNode = HEAD;
 - Variabel curNode diinisialisasi agar menunjuk ke node pertama, yaitu HEAD.
- while (curNode != null) { ... }
 - Perulangan akan berjalan selama curNode belum mencapai null.
 - null menandakan bahwa kita telah sampai di akhir linked list.
- System.out.print(curNode.getData() + " ");
 - Mencetak nilai data (yang disimpan pada setiap node) diikuti spasi.
- curNode = curNode.getNext();
 - Menggeser curNode ke node berikutnya, sehingga perulangan dapat terus bergerak maju sampai akhir list.



```

1  public class ListMain {
2
3      Run | Debug
4      public static void main(String[] args) {
5          StrukturList list = new StrukturList();
6          list.addTail(data:3);
7          list.addTail(data:4);
8          list.addTail(data:5);
9
10         System.out.println("Elemen :");
11         list.display();
12     }
13 }

```

Penjelasan:

➤ Deklarasi Kelas dan Method Main

- Kelas **ListMain** dideklarasikan sebagai kelas publik.
- Di dalamnya terdapat **method main** yang menjadi titik awal eksekusi program.

➤ Membuat Objek StrukturList

- Pada baris `StrukturList list = new StrukturList();`, dibuat objek `list` yang merupakan instansiasi dari kelas `StrukturList`.
- Objek ini akan menyimpan data dalam bentuk linked list.

➤ Menambahkan Elemen di Akhir List

- Tiga pemanggilan `list.addTail(3)`, `list.addTail(4)`, dan `list.addTail(5)` menambahkan node dengan nilai 3, 4, dan 5 di akhir list.
- Akibatnya, isi list secara berurutan adalah $3 \rightarrow 4 \rightarrow 5$.

➤ Menampilkan Elemen List

- `System.out.println("Elemen :");` mencetak teks “Elemen :” sebagai penanda.
- `list.display()` memanggil method yang akan mencetak semua data di dalam list, mulai dari node pertama hingga node terakhir.

Hasil Outputnya adalah:

```
PS D:\KuliahSMT4\Praktikum Pemrograman 1> & 'C:\Program Files\Java8\redhat.java\jdt_ws\Praktikum Pemrograman 1_d3fc7d1e\bin' 'List
Elemen :
3 4 5
PS D:\KuliahSMT4\Praktikum Pemrograman 1>
```

-Latihan 5

Latihan 5

Latihan ini akan memberikan implementasi operasi penambahan elemen list di awal/head. Operasi ini direpresentasikan dengan *procedure* **addHead** dengan parameter data yang akan ditambahkan. Tambahkan atribut **HEAD** dengan tipe data Node kelas **StrukturList**.

Algoritma addHead	Bahasa Pemrograman
<pre>procedure addHead(data: integer) deskripsi newNode ← new Node(data); IF(HEAD = null) THEN HEAD ← newNode ELSE newNode.next ← HEAD HEAD ← newNode ENDIF</pre>	<pre>public void addHead(int data) { Node newNode = new Node(data); if (isEmpty()) { HEAD = newNode; } else { newNode.setNext(HEAD); HEAD = newNode; } }</pre>

➤ Membuat Node Baru

- `Node newNode = new Node(data);`
Baris ini membuat objek node baru dengan nilai data yang diterima sebagai parameter.

➤ Cek Apakah List Kosong

- `if (isEmpty()) { ... } else { ... }`
Metode `isEmpty()` umumnya memeriksa apakah `HEAD == null`.
- Jika `HEAD == null`, berarti list masih kosong, sehingga `newNode` langsung dijadikan `HEAD`.
- Jika list tidak kosong, kita perlu menghubungkan node baru ke node yang sebelumnya menjadi head.

➤ Menautkan Node Baru di Depan

- `newNode.setNext(HEAD);`
Node baru (`newNode`) diarahkan untuk menunjuk ke node yang sebelumnya menjadi head.
- `HEAD = newNode;`
Setelah itu, head diperbarui sehingga mengarah ke `newNode`.

1. TES-1:

Latihan 4

Latihan ini akan memberikan penggunaan operasi penambahan elemen di akhir list dan kemudian menampilkan setiap elemen yang terdapat di list. Buatlah kelas **ListTest** berikut fungsi main untuk mengeksekusi program. Konversikan urutan instruksi berikut di bawah ini ke fungsi tersebut!

Urutan Instruksi	Program
1. Create list dengan keyword new 2. Tambah elemen 3 di akhir list 3. Tambah elemen 4 di akhir list 4. Tambah elemen 5 di akhir list 5. Tampilkan elemen list	<pre>public class ListMain { public static void main(String[] args) { StrukturList list = new StrukturList(); list.addTail(3); list.addTail(4); list.addTail(5); System.out.println("Elemen: "); list.display(); } }</pre>

Outputnya adalah: **Elemen: 3 4 5**

Lakukan seperti diatas dengan output elemen list seperti berikut:

a. 3 2 1

b. 1 4 5 7

```
TugasDanPertemuanCoding > TugasPertemuan3 > J Test1.java
1 public class Test1 {
2
3     Run | Debug
4     public static void main(String[] args) {
5         StrukturList list = new StrukturList();
6         list.addTail(data:3);
7         list.addTail(data:2);
8         list.addTail(data:1);
9
10        System.out.println("Elemen :");
11        list.display();
12
13        System.out.println();
14
15        StrukturList list2 = new StrukturList();
16        list2.addTail(data:1);
17        list2.addTail(data:4);
18        list2.addTail(data:5);
19        list2.addTail(data:7);
20
21
22        System.out.println("Elemen :");
23        list2.display();
24    }
25 }
26
27
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\KuliahSMT4\Praktikum Pemrograman 1> & 'C:\Program
n' 'Test1'
Elemen :
3 2 1
Elemen :
1 4 5 7
PS D:\KuliahSMT4\Praktikum Pemrograman 1>
```

Jawab:

Penjelasan:

Di Class test 1 di dalam program main, membuat struktur list baru Bernama list. Lalu list ditambah di bagian akhir nomor, dengan urutan 3, 2 1. Setelah itu diperlihatkan elemennya di bagian output

Begitu juga dengan struktur list 2, code ini berfungsi sama dengan code sebelumnya hanya saja dengan list yang berberda, yakni 1,4,5,7. Hasil Query akan seperti di gambar berikut:

2. TES-2 dan TES-3

Tes-2

Latihan ini akan memberikan penggunaan operasi penambahan elemen di awal list dan kemudian menampilkan setiap elemen yang terdapat di list. Konversikan urutan instruksi berikut di bawah ini ke dalam Bahasa pemrograman!

Urutan Instruksi	Output
1. Create list dengan keyword new 2. Tambah elemen 5 di awal list 3. Tambah elemen 4 di awal list 4. Tambah elemen 3 di awal list 5. Tampilkan elemen list	3 4 5

Tes-3

Lakukan seperti diatas dengan output elemen list seperti berikut:

- 3 2 1
- 1 4 5 7

Jawab:

-Tes-2

```
TugasDanPertemuanCoding > TugasPertemuan3 > J Test2.java > ...
1  public class Test2 {
    Run | Debug
2      public static void main(String[] args) {
3          StrukturList list = new StrukturList();
4          list.addHead(data:5);
5          list.addHead(data:4);
6          list.addHead(data:3);
7
8
9          System.out.println("Elemen :");
10         list.display();
11     }
12 }
13

PS D:\KuliahSMT4\Praktikum Pemrograman 1> & 'C:\P
n' 'Test2'
Elemen :
3 4 5
PS D:\KuliahSMT4\Praktikum Pemrograman 1>
```

-Penjelasan:

Di Dalam Class Test-2 di dalam program main, ini berfungsi untuk membuat struktur list baru dengan nama list. Setelah itu code ini akan menambahkan nilai di awal dengan urutan 5, 4, 3. Ketika di Run, hasilnya akan Seperti gambar berikutnya.

TES-3:

```
1 public class Test3 {
2
3     //Bagian A//
4     public static void main(String[] args) {
5         StrukturList list = new StrukturList();
6         list.addHead(data:1);
7         list.addHead(data:2);
8         list.addHead(data:3);
9
10
11         System.out.println("Elemen :");
12         list.display();
13
14     }
15     //Bagian B//
16
17     StrukturList list2 = new StrukturList();
18     list2.addHead(data:5);
19     list2.addHead(data:4);
20     list2.addHead(data:3);
21
22
23     System.out.println("Elemen :");
24     list2.display();
25 }
26
27 }
28
29
```

```
PS D:\KuliahsMT4\Praktikum Pemrograman 1> d:;
e' '-cp' 'C:\Users\ACER\AppData\Roaming\Code\U
1_d3fc7d1e\bin' 'Test3'
Elemen :
3 2 1
Elemen :
3 4 5
PS D:\KuliahsMT4\Praktikum Pemrograman 1> 
```

Penjelasan:

Hampir sama dengan code sebelumnya, code ini berfungsi untuk membuat list baru dengan nama list, hasil untuk query pertama adalah list dengan urutan 3, 2, 1.

Sementara untuk bagian B, hanya dibedakan dengan baris code yang membuat list baru bernama list2, yang akan menghasilkan list dengan urutan 5, 4, 3.

Hasil untuk kedua code ada di gambar berikutnya.